micomputer

CURSO PRACTICO
del computador personal
el micro y el minicomputador



PLANETA

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editado por
Planeta Colombiana Editorial, S. A., Bogotá

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Asesor técnico: Roberto Quiroga

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5o. - Barcelona-8

ISBN (Obra completa) 958-614-061-x
ISBN (Tomo I) 958-614-062-8
ISBN (Fascículo) 958-614-063-6

Impreso y encuadernado por Ediciones Lerner Ltda.
Calle 8ª A N° 68A-41-Bogotá D. E., Colombia
Impreso en Colombia-Printed in Colombia

Coordinación en Ecuador:
Editorial Planeta del Ecuador, S. A.
Av. Fco. de Orellana 1811 y 10 de agosto
Edificio El Cid
Quito - Ecuador

Coordinación en Venezuela:
Editorial Planeta Venezolana, S.A.
Quinta Toscanella, Calle Madrid - Las Mercedes
Caracas 1060, Venezuela

Distribución en Colombia:
Distribuidoras Unidas, S.A.
Transversal 93 No. 52-03
Bogotá D. E., Colombia

Distribución en Ecuador:
Distribuidora Muñoz Hnos., S. A.
General Aguirre 166 y 10 de Agosto
Quito - Ecuador

Distribución en Venezuela:
Distribuidora Continental, S. A.
Edificio Bloque de Armas
Final Avenida San Martín y final Avenida La Paz
Caracas - Venezuela

MI COMPUTER, *curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.

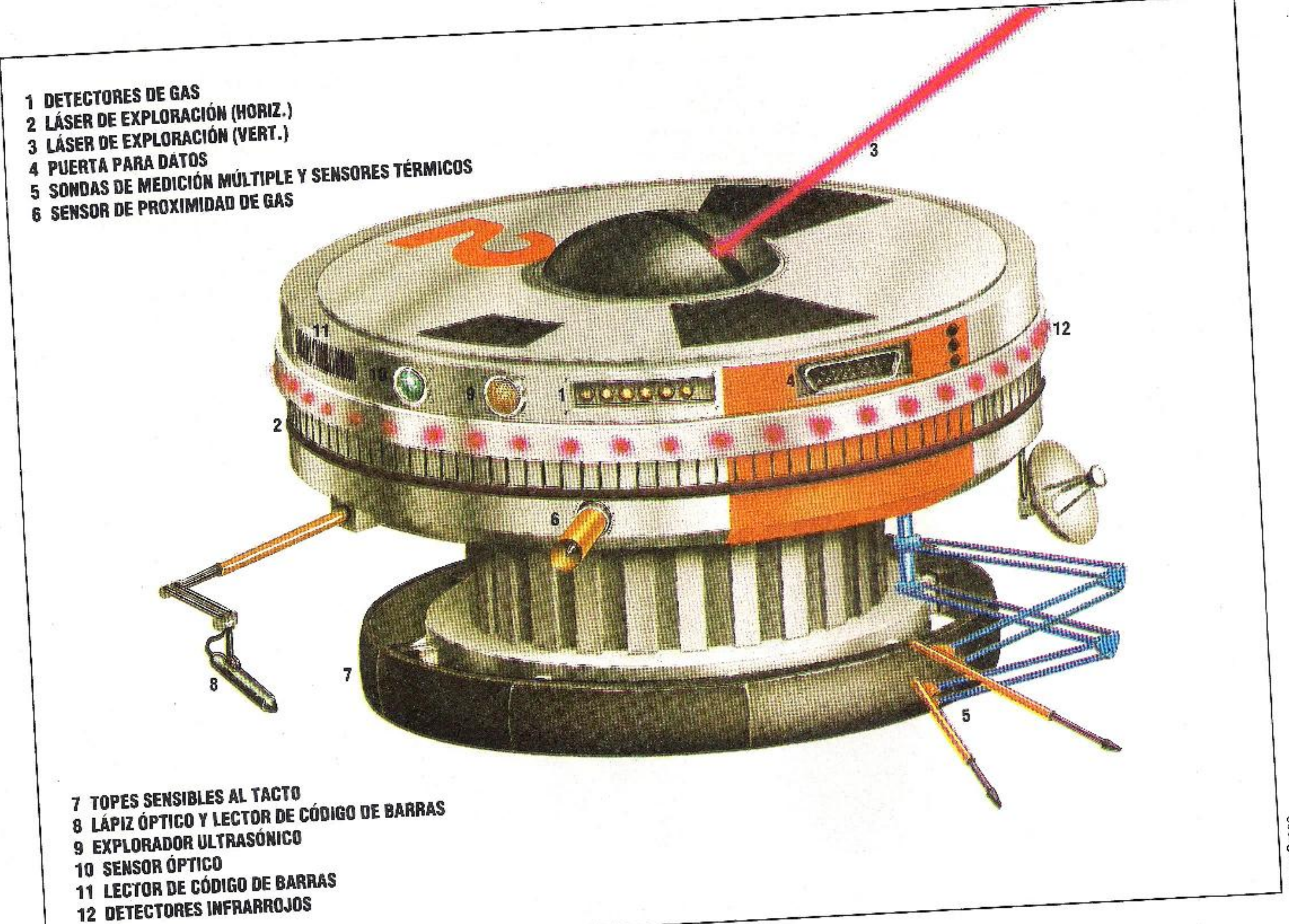
Planeta Colombiana Editorial, S. A., garantiza la publicación de la obra completa y se reserva el derecho de modificar el precio de venta del fascículo durante el transcurso de la obra, si las circunstancias de costos así lo exigieran.

Como la colección MI COMPUTER fue traducida originalmente en España, en el texto se usa la palabra ORDENADOR, que corresponde al término COMPUTADOR.



PLANETA

El mundo exterior



Sentidos y sensibilidad
El equipo sensorial que necesita un robot depende por completo de sus funciones; pero cuanto más generales sean los usos a que se destine el robot, más probable es que necesite más sensores. El robot de la ilustración ofrece ejemplos de la mayoría de los sensores posibles existentes, si bien no es probable que ningún robot pueda, él solo, incorporar una gama tan amplia

En esta ocasión estudiaremos de qué manera pueden los robots percibir lo que sucede a su alrededor

El sistema sensorial del hombre es algo que damos por sentado, pero una persona que carecíera de todos los sentidos estaría totalmente desvalida. Sin el sentido de la vista, se tropezaría con los objetos cuando se intentara caminar; sin el tacto, uno ni siquiera se enteraría de que había tropezado; la sordera total significaría que uno ni siquiera podría recibir una advertencia de que estaba a punto de llevarse un objeto por delante. De hecho, ni siquiera se podría caminar, porque para informarle al cerebro de la forma en que el cuerpo se está moviendo son necesarios los sentidos internos.

Hemos explicado cómo se puede desplazar un robot, pero debemos asimismo proporcionarle, antes de que pueda actuar con independencia, un sistema sensorial. Puede resultar apasionante tratar de diseñar un robot que posea *todos* los sentidos humanos: entonces podría percibir el mundo de forma muy similar a como lo percibimos nosotros. Por el momento, sin embargo, esto es imposible. Los temas relativos a la comprensión de lo que se ve y se habla son tan complejos que nos ocuparemos de ellos con profundidad en un futuro capítulo. Aquí nos concentraremos en sencillas aproximaciones a la vista y el oído que están muy por debajo del nivel de complejidad que poseen los humanos.

Es bastante sencillo hacer que un robot "vea" las cosas proporcionándole un sensor luminoso (por lo general, una célula fotoeléctrica) que produce un voltaje que varía con la cantidad de luz que recibe. Éste es un sensor de visión muy rudimentario, pero con él se pueden conseguir buenos efectos. Por ejemplo, se puede hacer que un robot se "dirija" hacia una luz brillante de modo muy similar a como se le hace seguir una línea (véase p. 1121). Esto se

puede utilizar para permitir que el robot localice un punto de energía para recargarse cuando se le agoten las pilas. (Tenga en cuenta que ello requerirá que el robot posea un sensor interno para controlar el estado de éstas: así "sabrá" cuándo están bajas.)

Esta sencilla célula fotoeléctrica puede permitir que un robot lleve a cabo numerosas tareas. Un robot instalado en una cadena de montaje podría estar capacitado para verificar si un componente está presente detectando la diferencia de brillo debida a la ausencia de aquél; esta tarea se puede simplificar disponiendo que la iluminación sea la adecuada para que se acentúe tal cambio. El robot podría detectar variaciones de color si se incorporaran tres células fotoeléctricas, respondiendo cada una de ellas a una luz de distinto color; el rojo, el verde y el azul cubrirían el espectro visible. Un robot de estas características se podría programar de modo que captara los ladrillos rojos de una pila que contuviera ladrillos de muchos colores diferentes. Esto produce la impresión, a partir de un sensor muy simple, de un comportamiento "inteligente".

Si al robot se le proporciona un micrófono, podrá "oír" señales acústicas. No "entenderá" lo que está oyendo, pero esto no tiene por qué ser importante: repitiendo varias veces un conjunto de instrucciones, el robot puede construir un "modelo" de sonido para cada una de ellas, que le permitirá comparar instrucciones nuevas con las oídas previamente. El número de instrucciones a las cuales podrá responder será limitado, pero podremos decirle que vaya "hacia adelante", etc., y obedecerá.

Un robot puede, asimismo, poseer un sencillo sentido del tacto. En su diseño se pueden incorporar microinterruptores de modo tal que establezcan una conexión eléctrica siempre que se les aplique una presión. Éstos carecen de la complejidad del sentido humano del tacto, pero aun así pueden ser muy útiles. Por ejemplo, sensores táctiles montados alrededor del borde de un robot móvil pueden permitirle responder inteligentemente a cualquier obstáculo: el robot será capaz de volverse hacia atrás ante la obstrucción y probar otra ruta. Sensores táctiles incorporados en una mano permitirán que éste "sepa" cuándo tiene algo a su alcance de modo que pueda responder en consecuencia.

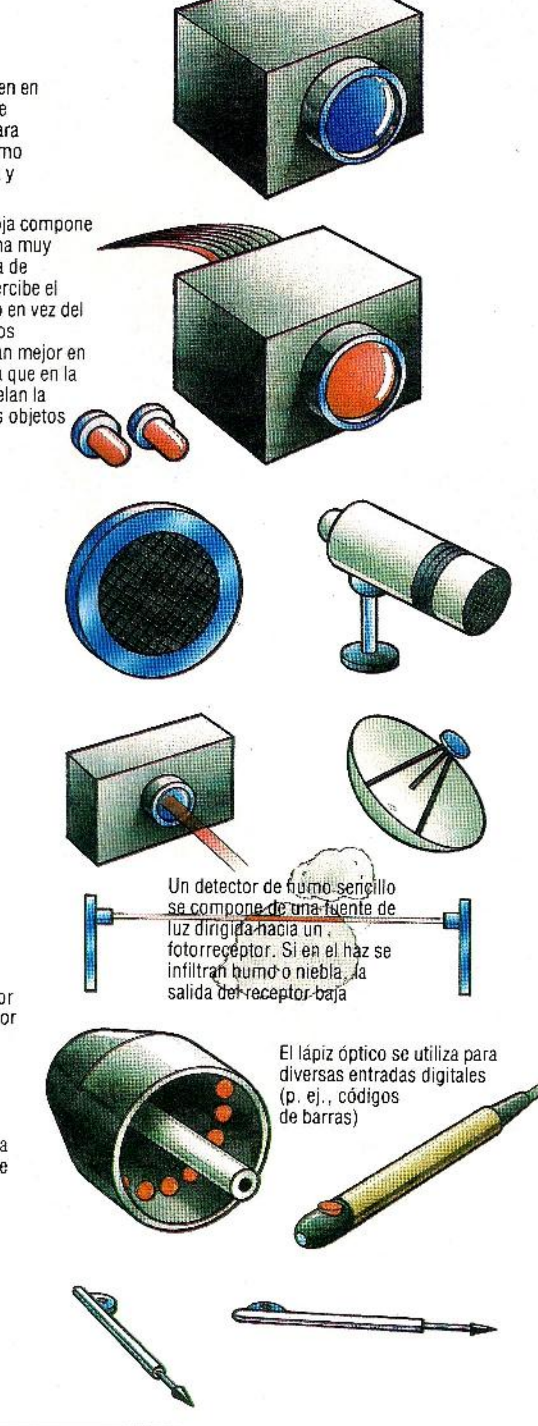
Se pueden utilizar detectores de humo o gas para darle al robot una especie de sentido de olfato. Los detectores de gas suelen utilizar un elemento sensorial (tal como un cable de platino) que responde ante la presencia de ciertos gases, alterando, por consiguiente, la corriente eléctrica que fluye a través del elemento. Los detectores de humo poseen dos cámaras: una cerrada, que actúa a modo de referencia o "control", y la otra abierta. Ambas cámaras contienen helio ionizado y la cantidad de partículas cargadas de la cámara abierta varía cuando hay humo. Un detector que cuente las partículas cargadas que hay en cada cámara registrará una diferencia entre las dos cuando haya humo.

Por el momento, parecería no haber forma alguna de dotar a un robot de sentido del gusto. No obstante, aplicando los métodos que hemos sugerido, al menos tendremos un robot que puede ver, oír, sentir y oler lo suficientemente bien como para detectar un incendio en el edificio, correr hacia las llamas, evitando los obstáculos que encontrará en su camino, y, en caso de que tuviera un extintor de incendios en su efector final ("mano"), sofocar el fuego con espuma.

Ganancia potencial

Pero limitando un robot al tipo de sentidos que poseemos los humanos estamos perdiendo mucho de su potencial. No existe ninguna razón por la cual se haya de restringir al robot a detectar cosas de la forma en que las detectamos nosotros. Un enfoque mejor sería el de considerar qué sentidos se le pueden proporcionar al robot y decidir si los mismos tienen algún uso práctico.

SENSORES



El sensor óptico es una cámara de televisión monocromática de baja resolución y exploración lenta. Produce una imagen en tonos grises que contiene información suficiente para tareas sencillas, tales como seguimiento de una línea y detección de bordes

La cámara infrarroja compone su imagen de forma muy similar a la cámara de televisión, pero percibe el espectro infrarrojo en vez del de la luz visible. Los infrarrojos penetran mejor en el humo y la niebla que en la luz, y también revelan la temperatura de los objetos

El ultrasonido es sonido de alta frecuencia, utilizado aquí para el cálculo de distancia de un objeto. El explorador se compone del emisor de ultrasonido y el receptor del micrófono direccional. Cuando el ultrasonido rebota en un objeto, la textura de la superficie reflectora distorsiona la forma de la onda del eco con una "firma" exclusiva e identificable

El explorador láser de baja potencia se utiliza para determinar con gran exactitud la dirección y la distancia. La luz láser se puede enfocar con gran precisión, lo que permite un examen detallado y exacto de los objetos cercanos

El detector de proximidad por gas se compone de un emisor de gas y un sensor de presión. El emisor arroja regularmente gas en la cámara, lo que produce un aumento conocido de la presión ambiental; si hubiera un objeto cerca de la boca de la cámara, afectaría a este aumento de presión de una forma detectable

Las sondas de medición múltiple (*tester*) permiten la medición de resistencia, capacidad eléctrica, voltaje y corriente; también pueden funcionar como sensor térmico

Kevin Jones

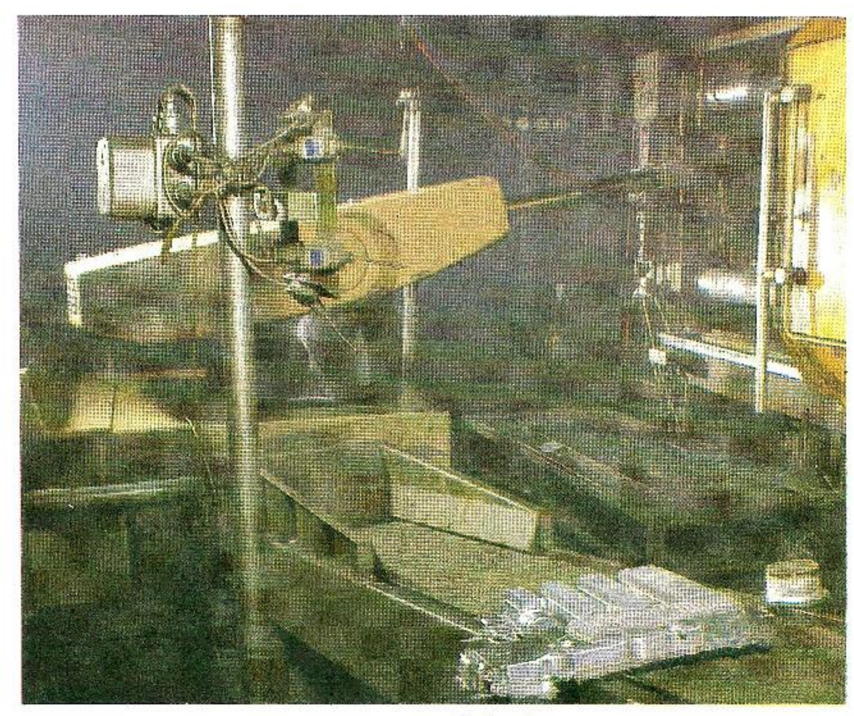
Un buen ejemplo de esto nos lo dan los brazos-robot. Vamos a suponer que deseamos que un robot recoja un objeto de un lugar y después lo coloque en algún otro sitio. Una forma de hacer esto es fijar topes alrededor del brazo, de modo que éste sólo pueda recorrer una distancia máxima preestablecida en cualquier dirección dada. El brazo se balancearía hasta llegar a los topes, en cuyo punto (si todo estuviera situado correctamente) la mano estaría directamente encima del objeto a coger. Después de asir el objeto, el brazo se balancearía en la dirección contraria hasta que otro tope le haría saber que debe soltarlo. Éste es un ejemplo sencillo y su uso se está reduciendo cada vez más, pero demuestra que a los robots se los puede dotar de sentidos de los cuales carecemos.

Tal vez sea más ilustrativo el ejemplo del empleo de la "vista" por parte de un robot. Los seres humanos sólo perciben la luz visible (una gran parte del espectro electromagnético es invisible al ojo humano), pero no hay ninguna razón por que el robot deba sufrir esta restricción. En vez de células fotoeléctricas se podrían instalar detectores infrarrojos; éstos permitirán medir la cantidad de calor generada por un objeto. Los robots industriales pueden hacer uso de estos detectores para alejarse de cualquier objeto cuya temperatura sea, por ejemplo, peligrosamente alta. Pero un robot también podría detectar la tibieza de un cuerpo humano, ¡de modo que usted podría programar su robot personal para que saliera corriendo a su encuentro cuando atravesara la puerta de su casa! También se puede hacer que los robots detecten campos magnéticos. Esto ya se ha analizado en relación a los robots que siguen una huella trazada en el suelo, pero esta facilidad también sería útil para aplicaciones en las cuales un robot hubiera de diferenciar entre materiales magnéticos y no magnéticos.

Los sensores de proximidad no tienen un equivalente exacto en el hombre; son sólo dispositivos que pueden detectar la cercanía de un objeto. Con este fin, los seres humanos se sirven de una combinación de vista y tacto, pero un sencillo sensor de proximidad es igualmente apto para el empleo en robots. Tales sensores trabajan de diversas maneras. Un tipo utiliza un chorro de aire arrojado a través de una boquilla; cualquier objeto que haya en el recorrido del chorro desviará el aire nuevamente hacia la boquilla. Esto crea una presión de retroceso que puede ser detectada por un transductor de presión, advirtiendo, por tanto, al robot de que hay algo cerca. Otro tipo se basa en el hecho de que un circuito eléctrico con un condensador cambiará su comportamiento si se está acercando a otro objeto. Una "fuga" entre el condensador y el objeto (que tendrá una capacidad eléctrica propia) le informará al robot que hay otro cuerpo próximo a él.

Transductores

Existen, asimismo, detectores ultrasónicos que funcionan emitiendo una señal ultrasónica y captando luego el eco producido por el objeto cercano. El tiempo transcurrido entre la señal y el eco proporciona una medida exacta de la distancia a la que se halla el objeto. Este método es similar al que utilizan los murciélagos para conocer su situación, y el principio también se emplea en algunas cámaras de enfoque automático.



Sin sentidos no hay sensación
Este brazo-robot industrial está sacando las piezas de hierro fundido de los moldes cuando aún están demasiado calientes para que las manos humanas las puedan tocar. El robot, por supuesto, es insensible al calor y, en consecuencia, realiza el trabajo más rápidamente

Los sensores láser son aún más sofisticados. Éstos dirigen un haz sobre un objeto, que entonces refleja la luz del láser de vuelta hacia el sensor. Mediante la comparación de ambos haces se puede determinar la distancia del objeto con una precisión asombrosa. Esta técnica se puede utilizar para grandes distancias. Durante el primer alunizaje de una nave tripulada se colocó en el satélite un reflector para permitir que un sensor láser pudiera medir la distancia exacta entre la Tierra y la Luna. Se afirma que el margen de error de esta forma de medición ¡es de 15 cm para una distancia de 384 400 km!

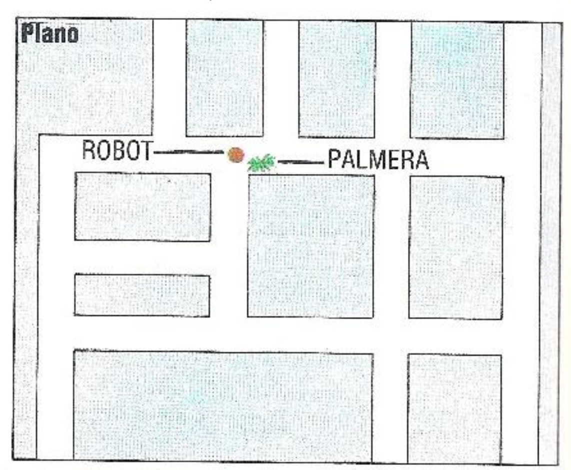
Los sensores de fuerza son un medio para obtener información táctil mediante medios más sofisticados que los microinterruptores mecánicos. Éstos operan midiendo el cambio producido en las propiedades eléctricas de un cristal piezoeléctrico cuando éste es sometido a presión, o calculando el cambio en la conductividad de gránulos de grafito de carbón bajo presión (utilizando una técnica idéntica a la empleada en el micrófono de carbón). Alternativamente, se pueden utilizar indicadores de tensión para medir fuerzas grandes detectando los cambios producidos en la resistencia eléctrica de un cable mientras el mismo es estirado.

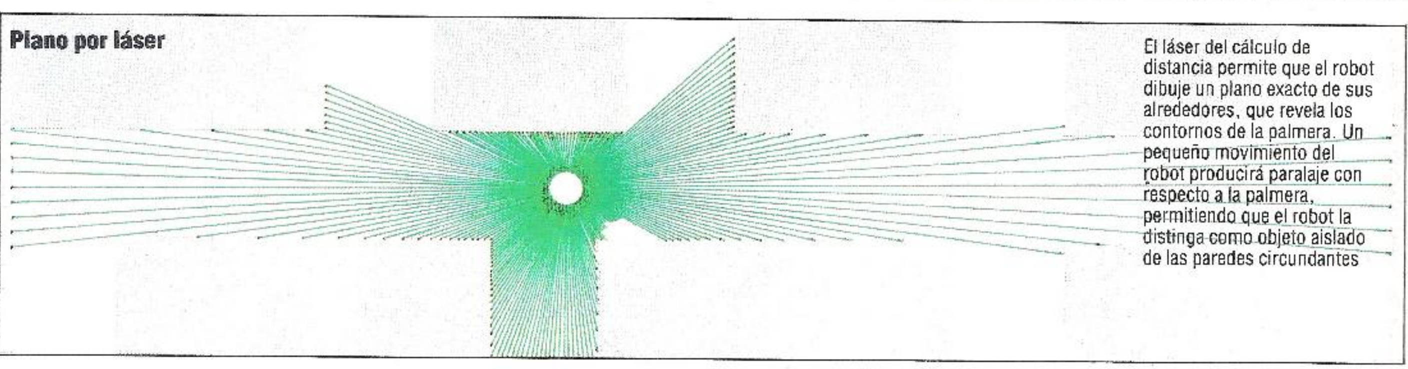
Estos sensores de robots se agrupan bajo la denominación común de *transductores*, dado que toman una medición de una forma (que puede ser luz, sonido o presión) y la convierten ("transducen") a otra que de alguna manera representa la medición original. En un robot controlado por ordenador, los transductores casi invariablemente convierten la medición en una señal eléctrica que puede ser binaria (es decir, la señal eléctrica está presente o no) o analógica (la señal varía en la medida en que la medición original cambia). En este último caso, la señal eléctrica se debe convertir a una forma que pueda entender el ordenador mediante el empleo de un convertidor A/D.

Es justo reconocer que los sentidos de un robot no son ni tan amplios ni tan eficaces como sus equivalentes humanos. Pero el robot posee más sentidos, y éstos se están perfeccionando día a día.

Sentidos mediante sensores

La percepción del mundo exterior es el mayor problema del robot, y aún lo es más cuanto mayor sea la gama y la complejidad de su equipo sensorial. No hay ningún sensor individual que proporcione una imagen completamente informativa, y algunos incluso parecen contradecirse. El nivel al cual el robot pueda integrar y comparar las entradas de sus varios sensores es lo que proporciona la medida de su "conciencia" externa.
En este ejemplo, el plano muestra que el robot se halla en un corredor cuyas paredes están pintadas de blanco; sólo hay una fuente de luz, de modo que la iluminación de una pared depende de su orientación. Cerca del robot hay una palmera

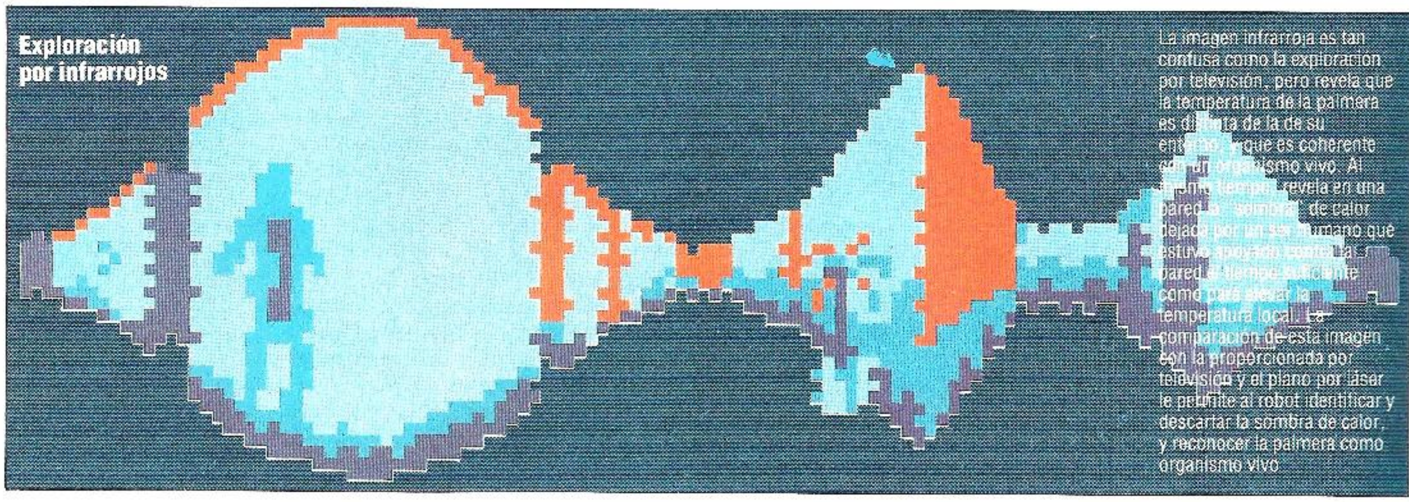




El láser del cálculo de distancia permite que el robot dibuje un plano exacto de sus alrededores, que revela los contornos de la palmera. Un pequeño movimiento del robot producirá paralaje con respecto a la palmera, permitiendo que el robot la distinga como objeto aislado de las paredes circundantes



La cámara de televisión proporciona una panorámica monocromática de baja resolución de los alrededores: ésta se puede mejorar mediante las técnicas de proceso de imágenes, para permitir la detección de bordes y la percepción de movimiento, pero contribuye poco a dar una imagen de la palmera. Sin embargo, cuando está integrada con otros exploradores, es un complemento útil.



La imagen infrarroja es tan confusa como la exploración por televisión, pero revela que la temperatura de la palmera es distinta de la de su entorno, lo que es coherente con un organismo vivo. Al mismo tiempo, revela en una pared la "sombra" de calor dejada por un ser humano que estuvo apoyado contra la pared el tiempo suficiente como para elevar la temperatura local. La comparación de esta imagen con la proporcionada por televisión y el plano por láser le permite al robot identificar y descartar la sombra de calor, y reconocer la palmera como organismo vivo

Steve Cross

Dígitos visualizados

En este capítulo agregamos dos visualizaciones de siete segmentos a nuestro sistema de la puerta para el usuario

Para visualizar dígitos hexadecimales se requieren cuatro bits (cuatro bits nos dan 16 permutaciones de ceros y unos). En consecuencia, cualquier número de ocho bits se puede representar utilizando dos dígitos hexadecimales: uno para los cuatro bits inferiores y otro para los cuatro superiores. A pesar de que cada visualización consta de siete segmentos LED, las diversas combinaciones de segmentos se pueden "activar" mediante cuatro líneas de entrada si se incorpora un circuito lógico de decodificación.

Los decodificadores son circuitos que traducen instrucciones del ordenador a sus periféricos en señales eléctricas, y viceversa. En nuestra serie sobre lógica construiremos nuestro propio circuito decodificador (véase p. 626), pero para este ejercicio podemos comprar un circuito lógico ya hecho. Éste es el chip 7447 de la lista de componentes.

El decodificador para cada visualización acepta cuatro líneas de entrada provenientes de la puerta para el usuario y, a través de una secuencia de puertas lógicas, proporciona siete salidas. El circuito lógico se ha diseñado de modo tal que si, pongamos por caso, las cuatro líneas de entrada fueran 0111, entonces se encenderían las barras correspondientes para visualizar el número 7 (0111 en binario equivale a 7 en hexadecimal). En la página 1167 podemos ver la tabla de verdad para ello.

Los dígitos hexadecimales mayores que nueve generalmente se representan mediante las siete primeras letras del alfabeto: de la A a la F. Se observará que el chip decodificador que estamos utilizando posee patrones un tanto extraños para representar estos dígitos. Es probable que estos patrones se puedan generar utilizando los circuitos lógicos requeridos para los dígitos del cero al nueve. Sería necesaria más lógica decodificadora para visualizar los seis últimos dígitos hexas de la forma alfabética más habitual, de modo que descartando la lógica extra y empleando símbolos distintos para estos dígitos se reduce la cantidad de puertas lógicas del decodificador, rebajando, por consiguiente, el costo que supone la fabricación del chip.

Una vez construido el circuito de visualización podemos visualizar continuamente el contenido del registro de datos de la puerta para el usuario en hexa, empleando las ocho líneas de entrada proporcionadas. Hay suficientes líneas disponibles para utilizar las dos visualizaciones simultáneamente, pero esto no sucede así en muchas aplicaciones, y varias visualizaciones de siete segmentos deben compartir las mismas líneas de datos. Para que cada visualización pueda mostrar distinta información al mismo tiempo, se utiliza una técnica llamada *multiplexión*. En esencia, las líneas de datos del decodificador de visualización se pasan rápidamente de una visualización a la siguiente, cambiando también de la forma correspondiente los datos presentes en las líneas. Si esto se realiza con rapidez suficiente, todas las visualizaciones multiplexadas de esta manera parecerán parpadear continuamente, visualizando cada una los datos que estén presentes en el instante en que se conecta a las líneas de datos.

Lista de componentes

Cantidad	Artículo
14	Resistencia 330 ohmios 0,4 vatios
2	BCD 7447 a decod. 7 segmentos
1	Visual. de 2 dígitos gemelos de ánodo común
2	Conector de chip DIL de 16 patillas
1	Con. minicon ángulo recto 12 vías
1	Ench. ángulo recto minicon 10 vías
	Cable plano de 8 vías*
	Cable plano de 7 vías*
	Cable pelado estañado*
1	Veroboard de 50 agujeros×36 franjas
1	Caja plástica de 116×61×36 mm

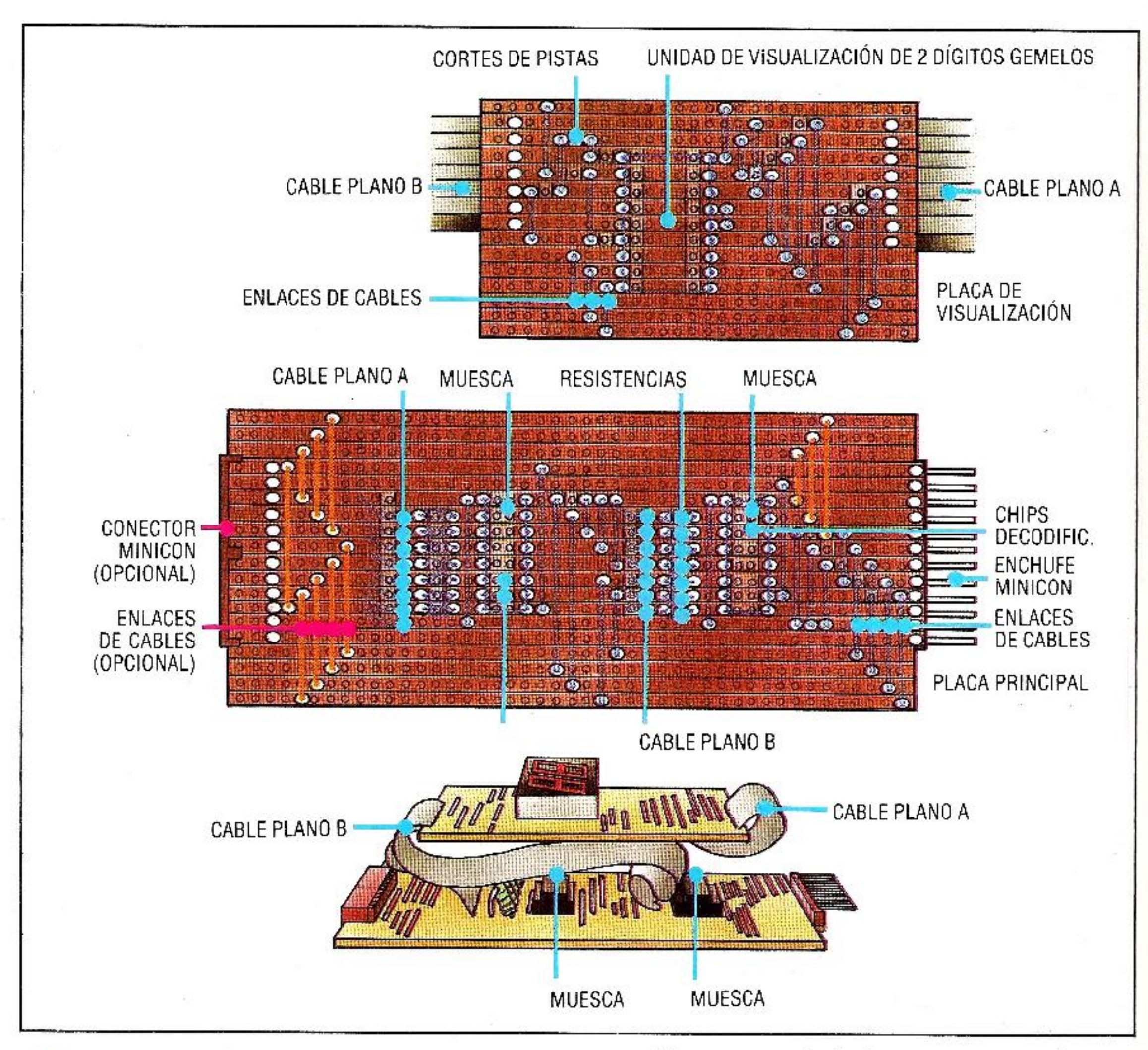
* Estos componentes pueden haberle sobrado de proyectos anteriores. El conector de 12 vías sólo es necesario si desea ampliar el bus del sistema

Podemos demostrar el principio de la multiplexión utilizando las dos visualizaciones de siete segmentos que estamos construyendo. Puesto que el decodificador de visualización representa al decimal 15 con un vacío, podemos emplear este número para borrar una visualización mientras se ilumina la otra. El siguiente programa, el ejecutarse, solicita un dígito a visualizar y después va mostrando el dígito en ambas visualizaciones al mismo tiempo. Sin embargo, se incluye una rutina que inserta una demora para retardar la oscilación entre las dos visualizaciones. La demora se inserta mientras se pulsa la barra espaciadora. Podemos ver, al ejecutar el programa y pulsar la barra, que en realidad el dígito salta una y otra vez de una visualización a la otra. Cuando se vuelve a liberar la barra espaciadora, se elimina la demora y el salto es más rápido.

```
10 REM MULTIPLEXION BBC
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=255
40 izquierda_vacio=15*16
50 derecha_vacio=15
55 :
60 REPEAT
70 INPUT"DATOS A MULTIPLEXAR";datos
80 ?REGDAT=datos+izquierda_vacio
90 PROCmas_despacio
100 ?REGDAT=datos*16+derecha_vacio
110 PROCmas_despacio
120 GOTO80
130 END
140 :
150 DEF PROCmas_despacio
155 REM SE ESTA PULSANDO LA BARRA ESPACIADORA?
160 IF INKEY (-99)=-1 THEN PROCdemora
170 ENDPROC
```



Planificándolo todo
Corte la veroboard en los dos tamaños necesarios (19 pistas de 46 agujeros; 15 pistas de 28 agujeros). Haga primero los cortes de pistas en ambas placas. Suelde los dos conectores de chip en su sitio, luego los enlaces de cables y las resistencias. Si no se requiere el conector para ampliación del bus, entonces omita los enlaces que en la ilustración aparecen en rojo. Coloque el enchufe minicon y el conector (opcional) en la placa principal, y suelde la unidad de visualización en su sitio (los puntos hacia el extremo del conector del tablero). Suelde los cables planos de conexión, de modo que vayan directamente de placa a placa sin retorcerse. Ahora enchufe los chips; asegúrese de que estén orientados tal como indica la ilustración




```
180 :
190 DEF PROCdemora
200 FOR I=1 TO 500:NEXT
210 ENDPROC

  10 REM MULTIPLEXION CBM 64
  30 RDD=56579:REGDAT=56577
  40 POKERDD,255
  50 IV=15*16:DV=15
  60 INPUT"DATOS A MULTIPLEXAR";DT
  70 POKEREGDAT,DT+IV
  80 GOSUB1000:REM MAS DESPACIO
  90 POKEREGDAT,DT*16+DV
 100 GOSUB1000:REM MAS DESPACIO
 110 GOTO70
 120 :
1000 REM S/R MAS DESPACIO
1010 GETA$
1020 IFA$=" "THENGOSUB2000:REM DEMORA
1030 RETURN
1999 :
2000 REM S/R DEMORA
2010 FORI=1 TO 250:NEXT
2020 RETURN
```

Una aplicación sencilla es utilizar las visualizaciones de siete segmentos gemelos como un contador hexadecimal. Ésta visualiza una cuenta del número de impulsos entrados en la puerta para el usuario desde un simple interruptor conectado a una línea de la puerta. A primera vista esta tarea parece trivial, hasta que uno comprende que se requieren las ocho líneas de la puerta para el usuario para la visualización, sin que quede ninguna para entrada. Si especificamos una de las líneas para entrada, pongamos por caso la línea 0, entonces el sistema de E/S del ordenador siempre retendrá esta línea alta, independientemente de qué número esté presente en el registro de datos. Si en éste fuera a colocarse 128 (10000000 en binario), entonces éste instantáneamente cambiaría a 129 (10000001) porque la línea 0 estaría retenida alta para entrada. Esto, como es obvio, daría valores de contador incorrectos en las visualizaciones. La solución estriba en aplicar una técnica similar a la multiplexión. Si utilizamos la línea 0 para que acepte entrada sólo durante un breve período y utilizamos todas las líneas para salida durante un período más largo, entonces en las visualizaciones parecerá brillar continuamente el valor correcto del contador, con apenas una oscilación del valor incorrecto producido por establecer momentáneamente la línea 0 para entrada.

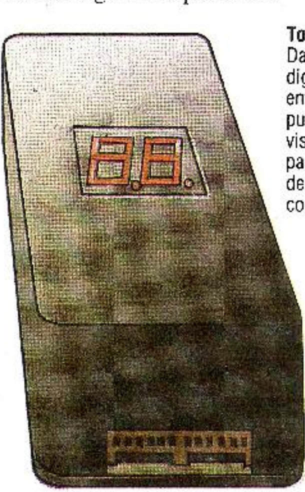
```
  10 REM CONTADOR BBC
  20 RDD=&FE62:REGDAT=&FE60
  50 contador=0
  55 :
  60 REPEAT
  70 PROCinput
  72 PROCsumar
  73 FORI=1TO40
  75 PROCvisualizacion
  77 NEXT I
  80 UNTIL contador>255
  90 END
 999 :
1000 DEF PROCsumar
```



```
1010 IF bandera=1 THEN contador=contador+1
1050 ENDPROC
1499 :
1500 DEF PROCinput
1510 ?RDD=254
1515 bandera=0
1520 IF(?REGDAT AND 1)=0 THEN bandera=1
1525 REPEAT UNTIL (?REGDAT AND 1)=1
1530 ENDPROC
1999 :
2000 DEF PROCvisualizacion
2010 ?RDD=255
2030 ?REGDAT=contador
2040 ENDPROC
```

```
  10 REM CONTADOR CBM 64
  30 RDD=56579:REGDAT=56577
  40 CC=0:REM INIC CONTADOR
  50 :
  60 GOSUB1000:REM ENTRADA
  70 GOSUB2000:REM SUMAR
  80 FOR I=1TO20
  90 GOSUB3000:REM VISUALIZACION
 100 NEXT I
 110 IF CC<255 THEN60
 120 END
 999 :
1000 REM S/R ENTRADA
1010 POKERDD,254
1020 FL=0
1030 IF (PEEK(REGDAT)AND 1)=0 THEN FL=1
1040 IF (PEEK(REGDAT)AND 1)<>1 THEN 1040
1050 RETURN
1999 :
2000 REM S/R SUMAR
2010 IF FL=1 THEN CC=CC+1
2020 RETURN
2999 :
3000 REM S/R VISUALIZACION
3010 POKERDD,255
3020 POKEREGDAT,CC
3030 RETURN
```

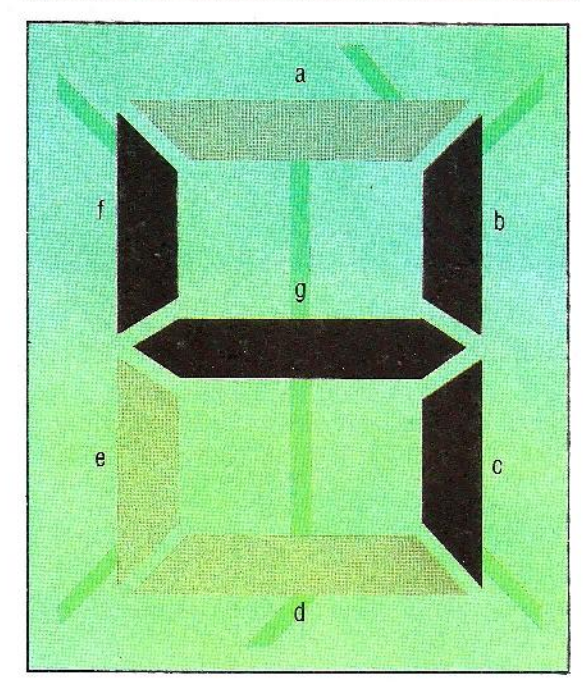
En cada ciclo del programa se utiliza un bucle FOR-...NEXT para repetir muchas veces la ejecución de la rutina en la cual se establecen todas las líneas en salida, para cada vez que se ejecuta la rutina donde se establece la línea 0 en entrada. En la versión para el Commodore 64, se ejecutan 20 rutinas de visualización por cada rutina de entrada. Esta proporción se incrementa a 40 en la versión para el BBC, debido a la mayor velocidad de ejecución del BBC Micro. Con estas proporciones, aún sigue detectándose un centelleo, pero si se incrementara esta proporción podría suceder que el tiempo de espera de una entrada se redujera tanto que las entradas llegaran a perderse.

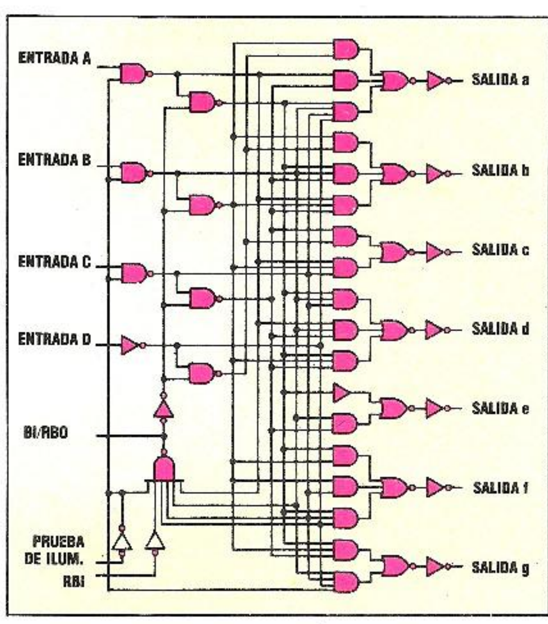


Todo empaquetado
Dado que cada visualización digital necesita un código de entrada de cuatro bits, se pueden activar dos visualizaciones desde la puerta para el usuario. Éstas se colocan dentro de una unidad compatible con interfaces y dispositivos

Decimal	Binario D3 D2 D1 D0	Salida a b c d e f g	Visual.
0	0 0 0 0	0000001	0
1	0 0 0 1	1001111	1
2	0 0 1 0	0010010	2
3	0 0 1 1	0000110	3
4	0 1 0 0	1001100	4
5	0 1 0 1	0100100	5
6	0 1 1 0	1100000	6
7	0 1 1 1	0001111	7
8	1 0 0 0	0000000	8
9	1 0 0 1	0001100	9
10	1 0 1 0	1110010	c
11	1 0 1 1	1100110	ɔ
12	1 1 0 0	1011100	u
13	1 1 0 1	0110100	ʕ
14	1 1 1 0	1110000	ɛ
15	1 1 1 1	1111111	

Calculando
La entrada a la visualización digital desde la puerta para el usuario es un número binario de cuatro bits. Relacionado con cada uno de los números del 0000 al 1111 hay un número exclusivo de siete bits, señalando cada uno de ellos el estado de uno de los siete segmentos de la visualización. Según este código de visualización, un bit cero significa que se ha de encender el segmento correspondiente, y un uno que el segmento correspondiente debe permanecer apagado

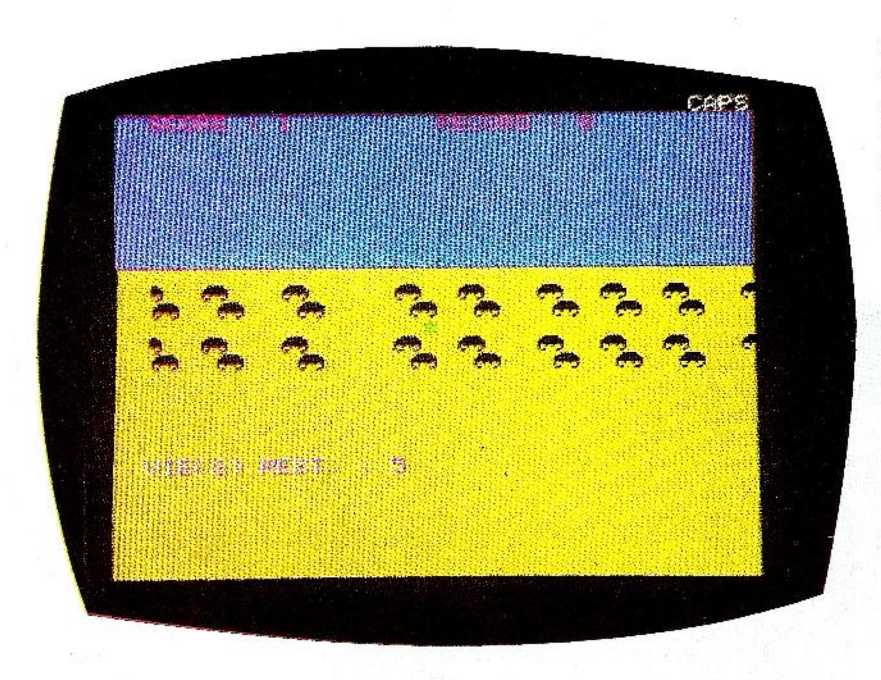




Decodificador/activador BCD 7447A a siete segmentos
El sistema de circuitos interno del chip muestra la sencillez esencial de su lógica: la entrada de cuatro bits se decodifica en salida para siete segmentos mediante puertas lógicas. La entrada de la prueba de iluminación enciende simultáneamente todos los segmentos para probar el chip

Cangrejos

Las leyes del mar revisadas y corregidas en función de la informática. He aquí un atractivo juego para el Oric Atmos



El usuario debe ayudar a una pobre tortuga a volver al mar, evitando a los voraces cangrejos que deambulan por la playa. Cada tortuga que alcance su propósito le proporciona un punto. Dispone de cinco itinerarios para intentar conseguir su puntuación máxima. Emplee las teclas W para avanzar y Z para retroceder.

```
10 REM **************
20 REM * CANGREJOS *
30 REM **************
35 POKE #26A,PEEK (#26A) OR 8
40 PRINT CHR$(17)
50 GOSUB 840
60 PLOT 2;20,"VIDA(S) REST. :"+STR$(NP)
90 AS=RIGHT$(AS,1)+LEFT$(A$,37)
100 BS=RIGHTS(B$,37)+LEFT$(BS,1)
120 PLOT 2,X1,AS
140 PLOT 2,X2,BS
160 PLOT 2,X3,A$
180 PLOT 2,X4,B$
190 DS=KEY$
200 PY=PY+(D$="W")-(DS="Z")
210 IF PY>16 THEN PY=16
220 IF PY=8 THEN 360
230 C=SCRN(PX,PY)
240 IF C<>32 AND C<>91 THEN 550
260 PLOT PX,YP,N$
280 PLOT PX,PY,P$
300 YP=PY
320 T=T+1
330 IF T>500 THEN 660
340 GOTO 60
360 PLOT PX,YP,N$
380 PLOT PX,PY,P$
420 GOSUB 1600
460 PY=16
470 YP=PY
480 S=S+1
500 PLOT 2,0,"PUNTOS :"+STR$(S)
520 PLOT 20,0,"RECORD :"+STRS(R)
530 GOSUB 1110
540 GOTO 60
550 NP=NP-1
570 PLOT PX,YP,N$
580 PLOT PX,PY,"*"
600 GOSUB 1500
610 IF NP=0 THEN 660
620 PY=16
630 YP=PY
640 GOSUB 1110
650 GOTO 60
660 CLS
665 INK 0
670 IF S>R THEN R=S
680 IF T<500 THEN 720
690 PLOT 8,6,"**TIEMPO TRANSCURRIDO**"
720 PLOT 13.10,"PUNTOS:"+STRS(S)
740 PLOT 12,14,"RECORD :"+STR$(R)
760 PLOT 11,20,"OTRA ?"
770 REPEAT
780 D$=KEY$
790 UNTIL DS=""
795 REPEAT
800 D$=KEYS
805 UNTIL D$<>""
310 IF D$ "N" THEN S0
315 CLS
820 PAPER 7
825 INK 0
830 PRINT CHRS(17)
835 END
840 CLS
850 PAPER 3
860 INK 2
870 RESTORE
880 FOR I=0 TO 23
890 READ A
900 POKE 46808+I,A
910 NEXT I
920 P$=CHR$(91)
930 N$=CHR$(32)
940 AS=""
950 BS=""
960 S=0
970 NP=5
980 PX=19
990 PY=16
1000 YP=PY
1010 X1=10
1020 X2=11
1030 X3=13
1040 X4=14
1050 T=0
1060 FOR I=1 TO 38
1070 READ A
1080 A$=A$+CHR$(A)
1090 NEXT I
1100 B$=A$
1110 X=INT(RND(1)*36)+1
1120 A$=RIGHT$(A$,X)+LEFT$(A$,38-X)
1140 PLOT 1,20,CHR$(5)
1150 PLOT 1,X1,CHR$(0)
1160 PLOT 1,X2,CHR$(0)
1170 PLOT 1,X3,CHR$(0)
1180 PLOT 1,X4,CHR$(0)
1190 PLOT 0,0,CHR$(18)
1200 PLOT 1,0,CHR$(5)
1210 FOR X=0 TO X1-2
1220 PLOT 0,X,CHR$(20)
1230 NEXT X
1240 RETURN
1500 ZAP
1510 RETURN
1520 WAIT 300
1530 RETURN
1600 PING
1610 FOR PY=PY TO 0 STEP-1
1620 PLOT PX,YP.N$
1630 PLOT PX,PY.P$
1635 WAIT 30
1640 YP=PY
1650 NEXT PY
1660 PLOT PX,YP,NS
1670 WAIT 100
1680 RETURN
2000 DATA 12,45,63,30,30.63.45,0
2010 DATA 7,15,31,31,18,16,12,0
2020 DATA 56,60,62,62,18,2,12,0
2030 DATA 32,92,93,32,32,92,93,32,32,32,
     92,93,32,32,32,32,32,92,93,32
2040 DATA 32,92,93,32,32,32,92,93,32,32,
     92,93,32,32,92,93,32,32,32
```



De compras

Establecemos una comparación entre algunos de los ordenadores más populares, destacando sus cualidades y sus puntos débiles

A la hora de adquirir una nueva máquina es importante considerar cuáles son exactamente las necesidades del futuro usuario: ¿desea un ordenador que se pueda ampliar mediante la adición de periféricos, memoria extra, etc., o puede permitirse tratarlo como un producto desechable, para vendérselo a otra persona cuando surja algo mejor?

La mayoría de los nuevos modelos ofrecen más facilidades que sus rivales más antiguos, incluyendo memorias más grandes, mejores versiones de BASIC, gráficos en resolución más alta y software incorporado. Pero las máquinas más antiguas, en especial aquellas de las cuales se han vendido grandes cantidades, poseen una ventaja fundamental: disponibilidad de software. Muchos compradores de ordenadores más nuevos habrán de esperar durante meses antes de disponer de una gran gama de software; y, en algunos casos, éste no aparecerá nunca. En este sentido, el Oric Atmos es un buen ejemplo. Esta versión mejorada del Oric-1 ha estado a la venta durante meses, pero los escritores de software se han mostrado reacios a producir material para el mismo. Como resultado de esto, las cifras de ventas de esta máquina han descendido drásticamente.

Los tres micros que están mejor servidos por las casas de software son el Sinclair Spectrum, el Commodore 64 y el BBC Micro. El Spectrum, en particular, es un clásico ejemplo de la forma en que la escritura de software creativo puede superar las limitaciones intrínsecas de una máquina: algunos de los programas para este micro se pueden comparar muy favorablemente con aquellos producidos para máquinas considerablemente más sofisticadas. Sin embargo, es poco probable que alguno de estos tres ordenadores se vendiera bien si se los lanzara al mercado de hoy en día: el Spectrum tiene un teclado sumamente pobre, el BASIC del Commodore 64 carece de las instrucciones que permitirían aprovechar al máximo el potencial de la máquina, y el BBC Micro tiene una memoria pequeña y su precio es excesivo para los estándares actuales.

La mayor parte de los micros más recientes poseen especificaciones más atractivas, pero carecen de la profundidad y amplitud de software. Cualquiera que adquiera una de estas máquinas está apostando a favor de que obtenga popularidad y, que, por tanto, convenza a quienes desarrollan software para que creen programas para la misma.

La tendencia principal en el caso de los ordenadores personales nuevos es ofrecer más por el mismo dinero. Los teclados de gran calidad, las memorias más grandes para el usuario (64 Kbytes o más) y los buenos gráficos en la actualidad ya son algo estándar. La calidad del intérprete de BASIC se ha mejorado considerablemente en máquinas como el Commodore Plus/4, Commodore 16, Sinclair QL y los micros MSX. El Amstrad incluye en su precio hasta una pantalla monocromática o en color.

Otra interesante tendencia actual es la inclusión de software "empaquetado" o gratuito. El Sinclair QL se suministra con cuatro programas de esta clase: un paquete de tratamiento de textos, hoja electrónica, base de datos y gráficos de gestión. El Commodore Plus/4 proporciona una gama similar, si bien los programas son menos sofisticados y, en realidad, requieren una unidad de disco para poder utilizarlos. Otros micros se concentran en los juegos. Con el Commodore 16 se suministran cuatro juegos, y hasta Sinclair ha empezado a proporcionar un paquete de seis juegos con su ya un tanto anticuado Spectrum.

Quien adquiera un micro nuevo deberá considerar también otros puntos. Algunas máquinas son más ampliables que otras, permitiendo la utilización de unidades de disco, impresoras, modems y otros periféricos. Algunos ordenadores aceptarán accesorios estándares, mientras que otros exigirán periféricos de su "propia marca", lo que limita las opciones del usuario. Un buen manual es esencial: algunas máquinas se entregan con manuales deficientes, que en lugar de aclarar las cosas lo que hacen es confundir. El comprador en cierne también deberá considerar el tipo de software disponible para cada máquina; por ejemplo, el BBC Micro posee una elevada proporción de software educativo, mientras que el Spectrum es una opción mejor si de juegos se trata.



Ian McKinnell

Para una decisión acertada
Comprar un ordenador debería ser tan fácil como adquirir un traje o un vestido. Sin embargo, el cúmulo de información técnica y la amplia gama de opciones que se le presentan al usuario convierten el acto de decidirse en una auténtica tómbola. Lo primordial antes de visitar la tienda de ordenadores es comparar ecuánime y desapasionadamente lo que usted en realidad necesita y la capacidad y características de las diferentes máquinas. Trate de decidir con antelación qué es lo que va a comprar, y deje que la "atracción" de la máquina escogida sea el último —nunca el primero— factor decisorio





Amstrad CPC 464

Amstrad es una empresa que posee una rica experiencia en el mercado de alta fidelidad, y ésta se refleja en el diseño compacto de su primer micro personal (véase p. 909). El hecho de que la máquina se suministre con una pantalla y con una grabadora de cassette incorporadas la hará muy atractiva para el usuario novel, y el Amstrad configura interfaces tanto para palanca de mando como Centronics. Existe, asimismo, una unidad de disco que se vende junto con el lenguaje LOGO y el sistema operativo CP/M. El Amstrad es una máquina buena y completa, con una resolución para gráficos máxima de 640 por 200 pixels en dos colores, con una "paleta" de 27 tonalidades y facilidades para sonido estéreo

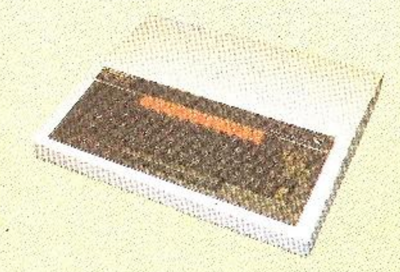
Sinclair QL

El QL se vende con 128 K de memoria, microdrives gemelos incorporados y cuatro programas de gestión empaquetados (véase p. 981). En vista de lo cual, parece ofrecer unas prestaciones notables; pero existen limitaciones. El teclado es decepcionante: es sólo una versión más sofisticada del teclado estilo membrana utilizado en el Spectrum; el BASIC es estructurado, pero contiene algunos errores y es sorprendentemente lento; la fiabilidad a largo plazo del microdrive es, asimismo, dudosa. Las facilidades de edición también son decepcionantes.

El QL soporta una resolución para gráficos de 512 por 256 pixels en cuatro colores, o de 256 por 256 en ocho. Cada pixel se puede colorear de forma individual, de modo que se utilizan 32 K de memoria sólo para manipular la pantalla. No se suministra interface para disco ni para cassette, pero el QL posee una interface para palanca de mando, otra para pantalla, una tercera para conexión en red y dos RS232. Es probable que uno de los principales reclamos de cara a las ventas sea el software empaquetado, escrito por Psion. Estos cuatro programas (tratamiento de textos, base de datos, hoja electrónica y programas para gráficos de gestión) son muy sofisticados en comparación con el software de otras máquinas personales. No obstante, se ven deslucidos por las limitaciones del hardware

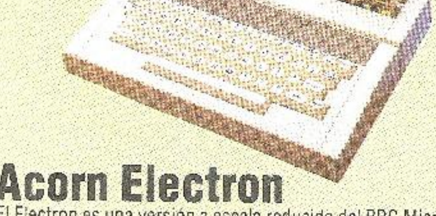
Sinclair Spectrum

El Spectrum se ha convertido en un éxito a pesar de sus limitaciones (véase p. 530). El teclado es muy pobre y el ordenador utiliza un sistema de "palabra-tecla" para la entrada de programas, lo que hace que la programación le resulte más sencilla al principiante pero que le plantee problemas al usuario más experimentado. La resolución de la pantalla es de 256 por 176 pixels con ocho colores, dos de los cuales se pueden utilizar en cualquier posición de carácter. La facilidad de sonido es prácticamente inexistente, con una única "voz" y un nivel de volumen virtualmente inaudible. El BASIC es aceptable, aunque algo lento, pero el manual cumple bien su función de enseñar el BASIC. El Spectrum carece por completo de interfaces estándares, si bien muchas firmas independientes han producido equipos de periféricos que simplemente se "enganchan" en la puerta para el usuario de la máquina. Recientemente Sinclair ha producido su propia Interface 1, que soporta microdrives, conexión en red y un enlace RS232. Ésta fue rápidamente seguida por la Interface 2, que le confirió a la máquina capacidad para utilizar software en cartucho. Aunque algo anticuada, la base de software del Spectrum lo convierte en una propuesta atractiva, en especial porque con la versión de 48 K se suministran de forma gratuita seis programas



BBC Modelo B

Esta máquina se utiliza bastante en los centros docentes; por este motivo puede resultarles familiar a los usuarios más jóvenes; existe una buena gama de software educativo (véase p. 929). Fabricado por Acorn, las especificaciones del BBC Micro son excelentes, con un BASIC "estructurado" muy veloz, excelente resolución para gráficos, buen sonido y una notable gama de interfaces, incluyendo Centronics, RS423, RGB, video compuesto, cuatro canales de A/D, una puerta para el usuario, bus de 1 MHz para accesorios y el "tubo", que permite conectar a la máquina un segundo procesador. También soporta unidades de disco y conexión en red. No obstante, la relativamente pequeña RAM para el usuario es devorada rápidamente por los gráficos, dejando entre 9 y 28 K para el usuario, dependiendo de la modalidad seleccionada. La utilización de un segundo procesador atenúa este problema, y la selección de la opción Z80 para segundo procesador permite la ejecución de software CP/M, haciendo al BBC apto para su empleo como máquina de gestión



Acorn Electron

El Electron es una versión a escala reducida del BBC Micro; posee el mismo y excelente BASIC estructurado, pero carece de la amplia gama de interfaces de que dispone el BBC (véase p. 929). El BASIC Electron trabaja a una velocidad inferior a la de la versión BBC, y el Electron no posee la modalidad 7 del BBC para gráficos de teletexto. Parte del software para el BBC es compatible con el Electron, mientras que existen otros programas que se han escrito especialmente para él. El Electron puede producir visualizaciones notables, con una resolución máxima para gráficos de 640 por 256 pixels, y una resolución máxima para textos de 32 líneas por 80 caracteres. Lamentablemente, esto reduce la cantidad de memoria disponible para el programador: del máximo de 32 K, al usuario apenas le quedan 9 K si selecciona la resolución más alta. El periférico Plus 1 proporciona las interfaces para impresora, palanca de mando y cartucho que no vienen con la máquina básica, pero hasta ahora no hay ninguna unidad de disco disponible

Para comprar un micro

La compra de un nuevo micro puede ser una fuente de confusión. Nuestro gráfico especifica las características más importantes de un micro y da perfiles comparativos para las máquinas incluidas en este capítulo. Todo lo que debe hacer es comparar sus necesidades con estos perfiles

Amstrad CPC 464

Pobre — Bueno

Precio
Memoria
Almac. de apoyo
Calidad teclado
Gráficos BASIC
Sonido BASIC
Editor BASIC
Facilidades BASIC
Calidad software
Cantidad software
Interfaces
Salida a pantalla

Sinclair QL

Pobre — Bueno

Precio
Memoria
Almac. de apoyo
Calidad teclado
Gráficos BASIC
Sonido BASIC
Editor BASIC
Facilidades BASIC
Calidad software
Cantidad software
Interfaces
Salida a pantalla

Sinclair Spectrum

Pobre — Bueno

Precio
Memoria
Almac. de apoyo
Calidad teclado
Gráficos BASIC
Sonido BASIC
Editor BASIC
Facilidades BASIC
Calidad software
Cantidad software
Interfaces
Salida a pantalla

BBC Modelo B

Pobre — Bueno

Precio
Memoria
Almac. de apoyo
Calidad teclado
Gráficos BASIC
Sonido BASIC
Editor BASIC
Facilidades BASIC
Calidad software
Cantidad software
Interfaces
Salida a pantalla

Acorn Electron

Pobre — Bueno

Precio
Memoria
Almac. de apoyo
Calidad teclado
Gráficos BASIC
Sonido BASIC
Editor BASIC
Facilidades BASIC
Calidad software
Cantidad software
Interfaces
Salida a pantalla





Commodore Plus/4

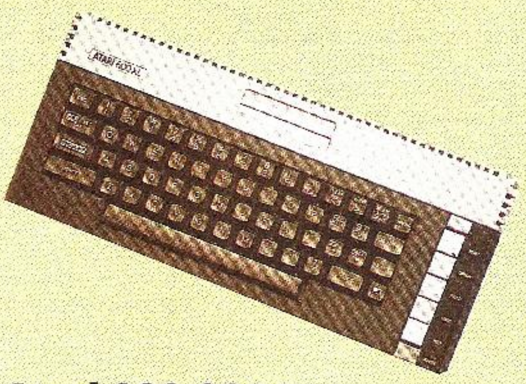
Ésta es la máquina que a la larga podría sustituir al Commodore 64. Ambos ordenadores poseen una resolución para gráficos similar de 320 por 200 pixels y 64 K de RAM, pero el Plus/4 puede visualizar 121 colores y tiene un BASIC muy mejorado, lo que le proporciona al usuario mayor control sobre la visualización en pantalla. En la modalidad de resolución más alta, sólo se pueden visualizar dos colores en cada posición de carácter, pero la selección de la modalidad de 160 por 200 pixels permite que sean cuatro por posición. El sonido no está del todo a la altura de los elevados estándares del C64, con un máximo de dos "voces"; sin embargo, su superior BASIC hace que la manipulación de sonido resulte mucho más fácil. En el nuevo modelo se ha omitido la facilidad para gráficos sprite del C64. El micro lleva incorporado un monitor de código máquina, así como cuatro pequeños programas "serios": tratamiento de textos, hoja electrónica, base de datos y programa para gráficos. Sin embargo, para aprovechar al máximo los mismos es necesaria una unidad de disco. El Plus/4, de acuerdo a la política de Commodore, exige su propia grabadora de cassette, pero éste no es del mismo modelo que utilizan el Vic y el Commodore 64. También se requieren palancas de mando especiales. El Plus/4 emplea las mismas impresoras y la misma lenta unidad de disco que el C64, si bien se está desarrollando una unidad más rápida

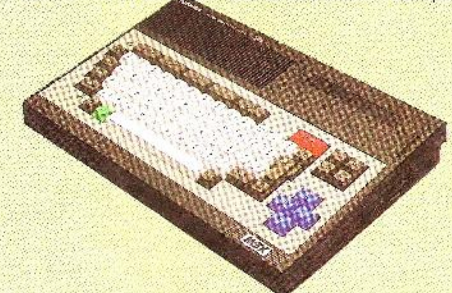
Commodore 64

Micro bien establecido en el mercado, con gran riqueza de software disponible, el 64 adolece de un BASIC pobre, que carece de instrucciones incorporadas para sacar partido de los excelentes gráficos y sonido (véase p. 490). La resolución máxima es de 320 por 200 pixels con 16 colores en pantalla, si bien en cada posición de carácter sólo se pueden visualizar dos colores. También posee sprites. A pesar de los 64 K de RAM, no hay más de 39 K disponibles para el usuario. El C64 exige una grabadora de cassette especial y, si se desea una ampliación, son necesarias impresoras y unidades de disco de la misma marca. La unidad de disco es bastante lenta en operación y muy sujeta a errores

Commodore 16

Diseñado para reemplazar al ya superado Vic-20, el Commodore 16 se suministra en un "paquete de iniciación" que contiene grabadora de cassette, cinta y libro para aprendizaje de BASIC y cuatro programas "recreativos". Si bien la carcasa le proporciona un aspecto similar al Commodore 64 y al Vic-20, por dentro la máquina se aproxima mucho más al Plus/4, ya que utiliza el mismo BASIC y el monitor de código máquina. En uso normal, quedan 12 K para empleo en BASIC, si bien los gráficos en alta resolución reducirán esta cantidad a unos insuficientes 2 K. Esta escasa asignación de memoria significa que la mayor parte del software para el 16 se encuentra en cartucho. De momento es muy poco el software que ha aparecido, pero los programas que se proporcionan con ella incluyen dos juegos recreativos, un programa de ajedrez que a la vez lo enseña, y un programa para diseñar gráficos. El C16 ofrece una buena relación calidad-precio, en especial para el novato, pero la pequeña cantidad de memoria libre podría plantearles problemas a los escritores de software





MSX Estándar

El MSX es un "estándar mínimo" y muchos fabricantes ofrecerán más que las especificaciones básicas, si bien ninguna de las mejoras afectará a la compatibilidad. La máquina de nuestra ilustración es el Toshiba HX-10 (véase p. 1149). El estándar MSX especifica una versión particularmente buena de BASIC, que incluye instrucciones para gráficos, sonido y tratamiento de eventos muy fáciles de utilizar, así como un buen editor. Las máquinas MSX poseen teclas de función, que se pueden programar para 10 funciones o instrucciones diferentes. La visualización da una resolución para gráficos de 256 por 192 pixels en 16 colores; también hay 32 sprites disponibles. Para hacer frente a la visualización, de los 80 K de RAM hay 16 reservados para la manipulación de pantalla. De los 64 restantes, el programador puede emplear 28 K; para acceder a los demás es necesario utilizar código máquina o bien una unidad de disco

Atari 600/800 XL

Estas máquinas son versiones mejoradas de la antigua serie Atari 400/800 (véase p. 669). Esto significa que hay una amplia gama de software disponible, si bien los ordenadores ahora parecen un poco anticuados. El Atari 600XL gradualmente está dejando de producirse, pero continúa siendo una buena adquisición mientras existan stocks, dado que sus 16 K de RAM se pueden ampliar a 64, convirtiéndolo de hecho en un 800 XL. La resolución máxima para gráficos es de 320 por 192 pixels, si bien en esta modalidad sólo se pueden visualizar dos colores. La selección de una resolución más baja otorga 16 colores en 16 tonalidades diferentes. Otras características notables son un sonido y unos gráficos sprite sobresalientes, aunque el BASIC que utiliza Atari en la actualidad está algo desfasado. Las máquinas Atari exigen una reproductora de cassette exclusiva, que implica un gasto adicional. Es de lamentar que, junto con los micros, no se suministre un manual completo; éste se debe adquirir por separado. Los periféricos Atari no son estándares, pero suele ser muy fácil conseguirlos y su precio es razonable

Descenso de los precios
Hemos preferido no incluir los precios de los ordenadores, dado que están variando de manera constante. Se advierte una significativa tendencia al descenso en los últimos meses, siendo éste, a veces, drástico

Commodore Plus/4 — Pobre / Bueno
- Precio
- Memoria
- Almac. de apoyo
- Calidad teclado
- Gráficos BASIC
- Sonido BASIC
- Editor BASIC
- Facilidades BASIC
- Calidad software
- Cantidad software
- Interfaces
- Salida a pantalla

Commodore 64 — Pobre / Bueno
- Precio
- Memoria
- Almac. de apoyo
- Calidad teclado
- Gráficos BASIC
- Sonido BASIC
- Editor BASIC
- Facilidades BASIC
- Calidad software
- Cantidad software
- Interfaces
- Salida a pantalla

Commodore 16 — Pobre / Bueno
- Precio
- Memoria
- Almac. de apoyo
- Calidad teclado
- Gráficos BASIC
- Sonido BASIC
- Editor BASIC
- Facilidades BASIC
- Calidad software
- Cantidad software
- Interfaces
- Salida a pantalla

Atari 600/800 XL — Pobre / Bueno
- Precio
- Memoria
- Almac. de apoyo
- Calidad teclado
- Gráficos BASIC
- Sonido BASIC
- Editor BASIC
- Facilidades BASIC
- Calidad software
- Cantidad software
- Interfaces
- Salida a pantalla

MSX Estándar — Pobre / Bueno
- Precio
- Memoria
- Almac. de apoyo
- Calidad teclado
- Gráficos BASIC
- Sonido BASIC
- Editor BASIC
- Facilidades BASIC
- Calidad software
- Cantidad software
- Interfaces
- Salida a pantalla

Máquina de calcular

He aquí una serie cuya finalidad es ser una guía práctica en el uso de hojas electrónicas basadas en cassette

Existen en el mercado varios paquetes de modelos financieros, basados en cassette, para ordenadores personales populares como el Sinclair Spectrum, Commodore 64 y BBC Modelo B. Esta serie de capítulos pretende proporcionar una guía práctica, paso a paso, del empleo de tales paquetes de hoja electrónica para numerosas aplicaciones cotidianas, entre las que se incluyen presupuesto doméstico, cálculo de la incidencia que podría tener cualquier aumento de los tipos de interés en los pagos de hipotecas, y comparación de los valores relativos de alquilar o comprar artículos para el hogar.

El corazón de todos los paquetes de hoja electrónica es una "hoja de trabajo" electrónica que se divide en filas y columnas (parecida a una gran hoja pautada de papel cuadriculado para gráficas y diseño). La pantalla de televisión (o de ordenador) actúa como una ventana móvil que puede visualizar cualquier porción de esta hoja (que es mayor que las cuatro o cinco columnas y las 10 o 15 filas que aparecen en pantalla en un momento dado).

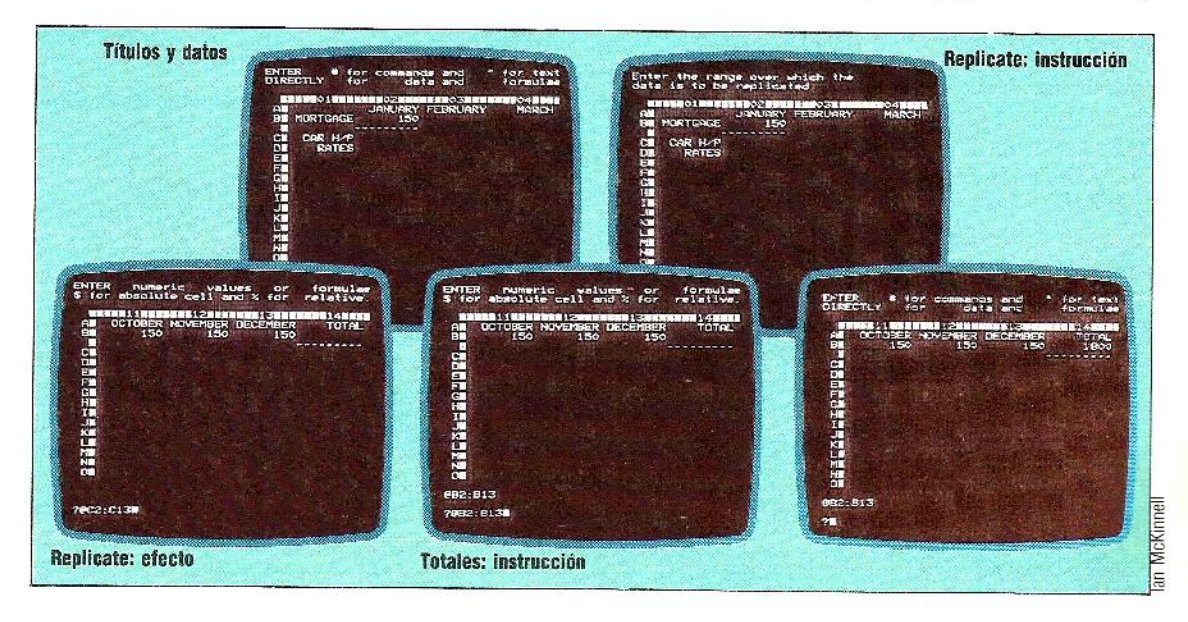
La intersección de una columna y una fila se denomina *celda*: cada celda puede contener números, texto o fórmulas. La hoja electrónica utiliza un cursor (normalmente un bloque realzado) que se puede desplazar a través de ella mediante las teclas para control del cursor. El programa supone que toda entrada de datos desde el teclado está destinada a la celda que en ese momento ocupa el cursor.

Éste es un esquema muy general del diseño básico de una hoja electrónica. En este primer capítulo nos concentraremos en un paquete llamado *Vu-Calc*; lo comercializa Psion y se vende tanto para el Spectrum como para el BBC Modelo B. Aquí vamos a considerar la versión para el BBC, que es casi idéntica (a excepción de ciertas diferencias, irrelevantes pero molestas, en los nombres de algunas instrucciones) a la versión para el Spectrum.

El *Vu-Calc* demuestra lo flexible y lo útil que puede ser una hoja electrónica sencilla. No posee ninguna de las características más sofisticadas de paquetes de modelización como el *Lotus 1-2-3* (véase p. 1124). Uno no puede dividir la pantalla ni horizontal ni verticalmente para mostrar distintas porciones de la hoja de trabajo al mismo tiempo (una facilidad de que disponen los paquetes más "serios"), y tampoco se dispone de la cantidad de facilidades que ofrecen las hojas electrónicas diseñadas con fines de gestión. Pero se puede utilizar el *Vu-Calc* para construir algunos modelos muy útiles, que se pueden guardar en cassette, junto con sus datos, para referencia posterior.

La versión del *Vuc-Calc* para el BBC Micro posee un máximo de 28 columnas (numeradas del 1 en adelante) y 52 filas (etiquetadas alfabéticamente, con las filas después de la "Z" etiquetadas con letras dobles: "AA", "BB", etc.). Este tamaño no es muy grande de acuerdo a los estándares de hojas electrónicas, pero el límite obedece a la más bien escasa memoria de 32 K del BBC Micro.

Una de las aplicaciones más útiles para una hoja electrónica de ordenador personal es el presupuesto doméstico anual, que posee asimismo el mérito de ser un modelo de construcción relativamente simple. Una vez construido, un modelo de este tipo permite ver a simple vista qué impacto tendrá el aumento de cualquier gasto (p. ej., una cuenta del teléfono excesivamente elevada o un viaje al extranjero que no estaba previsto) sobre el superávit que se esperaba (en el supuesto, claro, de que esperara alguno). En otras palabras, una vez que el modelo está construido, se puede jugar con los datos y



ver cómo se reflejará el efecto de una alteración a través de la hoja electrónica como un todo.

Construyendo un modelo

El primer paso para construir tal modelo consiste en escribir en un trozo de papel una lista de todos los gastos domésticos que se le ocurran. La siguiente tarea es anotar debajo de cada título la cifra mensual esperada. Es aquí, en el punto de calcular o estimar la cantidad mensual y entrarla en el modelo, donde se hacen evidentes los puntos fuertes del *Vu-Calc*. Las categorías de gastos se digitan una tras otra a lo largo de la primera columna, y cada columna subsiguiente se etiqueta "Ene", "Feb", "Mar", etc. El *Vu-Calc* exige que todas las entradas de texto vayan precedidas por comillas, pero en la hoja electrónica visualizada este signo de puntuación desaparece. Las primeras filas y columnas de nuestro modelo, por consiguiente, podrían ser así:

	1	2	3	4
A		ENE	FEB	MAR
B	Coche			
C	Hipoteca	25 000		
D	Impuestos			

Esto nos proporciona la "forma" básica del modelo. Ahora necesitamos entrar los valores correspondientes, y para hacer esto utilizamos la instrucción REPLICATE del *Vu-Calc*.

El *Vu-Calc* posee una cantidad limitada de instrucciones, todas las cuales llevan el prefijo de un signo #. (Cuando se entra # en una celda vacía, el programa pasa a la "modalidad instrucción" y espera a que se le diga cuál es la instrucción que se desea emplear.) La más útil de estas instrucciones es REPLICATE, porque la parte más aburrida de la construcción de un modelo es la necesidad de digitar todos los valores que utilizará el mismo. REPLICATE es básicamente un dispositivo ahorrador de trabajo que permite entrar el mismo dato en muchas celdas diferentes de forma simultánea.

Una parte común del modelo de presupuesto/gastos domésticos son los gastos mensuales fijos, como los impuestos, la hipoteca o el alquiler. Si la hipoteca es, por ejemplo, 25 000 ptas por mes, se utilizará la instrucción REPLICATE para insertar esta cantidad en las 12 celdas adecuadas.

Cuando se invoca la instrucción REPLICATE, digitando #R, en la parte superior de la pantalla, por encima del modelo propiamente dicho, aparece la siguiente línea de indicación:

Replicate - Enter the cell to replicate, RETURN for the current cell. (Replicate - Entre la celda a reproducir, RETURN para la celda en curso.)

Ésta es una indicación para que se especifiquen las celdas que se han de copiar (observe que puede copiar una sola celda, no un bloque entero de celdas; ésta es una de las limitaciones más evidentes del *Vu-Calc*). La celda se especifica mediante sus coordenadas, con la letra de la fila en primer lugar, seguida del número de columna; por ejemplo: C2. Después de entrada la celda, la línea indicadora le pide que:

Enter the range over which the data is to be replicated. (Entre la serie en la cual se ha de reproducir el dato.)

Una serie de celdas se indica en *Vu-Calc* especificando la primera celda (o la situada más a la izquierda) de la serie y la celda situada más abajo y más a la derecha de la serie. (Imagine que el bloque de celdas es como un recuadro: hay que decirle al programa las coordenadas de las esquinas superior izquierda e inferior derecha del recuadro.)

En nuestro ejemplo, deseamos decirle al programa que coloque 25 000 en la serie de celdas de la Fila C, desde C3 hasta C13 (de la columna etiquetada "Feb" a la etiquetada "Dic"). Para esto el formato es #R,C2,C3:C13. Con esto se rellena automáticamente cada celda, casi al instante, con el valor 25 000. (El verdadero poder de la instrucción REPLICATE es copiar fórmulas de una celda a otra, pero ésta es un área bastante especializada, puesto que tales fórmulas pueden ser "relativas" respecto a una celda determinada o bien "absolutas", distinción y tema estos que abordaremos pronto.)

Ahora trabajamos con todos nuestros títulos de gastos de la misma forma. Observe que si decide que en según qué meses la cantidad para un gasto determinado debe ser mayor o menor que el valor estándar, puede simplemente mover el cursor hasta esa celda y entrar un nuevo valor. Éste se escribirá de inmediato sobre la cifra anterior.

En nuestro modelo, la columna 14 será la columna de totales anuales. Tendría poco sentido utilizar una hoja electrónica si se tuviera que usar una calculadora para ir sumando los gastos de cada mes, de modo que el *Vu-Calc* se puede utilizar para sumar todos los valores de cualquier fila o columna. La instrucción @ indica que se desea sumar los valores que contiene una serie de celdas; esta serie se especifica de la misma forma que antes. Por lo tanto, para totalizar los pagos de hipoteca de todo el año y colocar el resultado en la celda C14, sencillamente se pone el cursor en C14 y se digita @C2:C13. El resultado (300 000 en este caso) se visualiza inmediatamente.

En la celda C14 podríamos igualmente haber colocado una fórmula; en vez de sumar todos los valores podríamos haber entrado C2*12. *Vu-Calc* habría tomado esto como una fórmula, dado que la C no iba precedida por comillas, y la hubiera ejecutado de inmediato, dando el resultado 300 000. Esto sirve para ilustrar que a menudo hay más de una manera de conseguir un resultado determinado.

En el próximo capítulo del curso analizaremos cómo los gastos pueden "crecer" en función de porcentajes fijos, y cómo se pueden reproducir fórmulas de referencia de celdas relativas y absolutas.

Desplegando la hoja

Commodore 64

Busicalc: cassette/disco por Supersoft, Canning Road, Harrow HA3 7SJ, Gran Bretaña.

Insta-Calc Graphic: cartucho/disco por Dataview Wordcraft Ltd., Radix House, East Street, Colchester CO1 2XB, Gran Bretaña.

BBC Micro

Vu-Calc: cassette por Psion Ltd., 2 Hunstworth Mews, Gloucester Place, London NW1 6DD, G.B.

Spectrum

Vu-Calc: cassette por Psion Ltd., Gran Bretaña.

Vencedor del dragón

Continuando con el estudio de los sprites en LOGO, desarrollaremos el algoritmo de la "curva de persecución"

Aquí ofrecemos los procedimientos para un juego que utiliza sprites de LOGO. El jugador controla a un dragón que intenta llegar hasta una ciudad y destruirla. La defensa de la ciudad está en manos de un caballero volador (controlado por el ordenador), quien intentará matar al dragón. El jugador controla la dirección de movimiento del animal mediante la palanca de mando. Si se burla al caballero y logra acercarse a la ciudad, ésta se verá envuelta en llamas debido al fuego exhalado por el dragón.

Para ejecutar el juego habrá de leerse el archivo SPRITES, definir sus formas, entrar los procedimientos y luego digitar JUEGO. Después de llevar a cabo diversas tareas de preparación, el procedimiento JUEGO llama a JUGAR, que es el procedimiento central. JUGAR hace mover al dragón y al caballero y verifica si el monstruo ha llegado a la ciudad o si el caballero le ha acertado. Los restantes procedimientos realizan otras partes de JUGAR.

Las instrucciones de color disponibles son muy directas. Para establecer el color de fondo utilice BACKGROUND seguido de un número de color, y para establecer el color de un sprite (y el color de la línea que trazará cuando tenga bajado su lápiz) utilice PENCOLOR. A los números de color se les asignan nombres en INIC.VARIABLES, de modo que podamos especificar colores mediante nombres, utilizando instrucciones tales como PENCOLOR :ROJO.

En el procedimiento JUGAR, la línea:

```
SI ACERTADO? THEN DRAGON.DESTRUIDO
```

se utiliza para comprobar si el caballero ha hecho blanco en el animal. El procedimiento ACERTADO? ilustra la forma en que podemos escribir en LOGO nuestras instrucciones de decisión. Devuelve un valor "TRUE (verdadero) o "FALSE (falso), y éste se emplea como una entrada para las sentencias IF. El resultado "TRUE haría entonces que se llevara a cabo la acción condicionada a la pregunta.

ACERTADO? utiliza un procedimiento del archivo

Caballeros y dragones



El juego en acción



La ciudad en llamas



El dragón vencido



Ian McKinnell

Caballero contra dragón

```
TO JUEGO
  INIC. VARIABLES
  PREPARAR.PANTALLA
  JUGAR
END

TO INIC.VARIABLES
  MAKE"LLAMA1
  MAKE"LLAMA1 2
  MAKE"DRAGON 3
  MAKE"CABALLERO 4
  MAKE"CIUDAD5
  MAKE"CIUDAD1 6
  MAKE"ROJO 2
  MAKE"NEGRO 0
  MAKE"AZUL 6
  MAKE"NARANJA 8
  MAKE"AMARILLO 7
END

TO PREPARAR.PANTALLA
  DRAW
  FULLSCREEN
  BACKGROUND :AZUL
  TELL 0
  PU
  HT
  TELL :LLAMA1
  HT
  HT
  TELL :LLAMA
  HT
  POSITION :DRAGON 100 100 :ROJO
  BIGX BIGY
  POSITION :CIUDAD (-52)(-80) :NEGRO
  BIGX BIGY
  POSITION :CIUDAD1 (-100)(-80) :NEGRO
  BIGX BIGY
  TELL 4
  PU
  POSITION :CABALLERO (100)(-100)
    :AMARILLO
  SMALLX SMALLY
END

TO JUGAR
  MOVIMIENTO.DRAGON
  IF DISTANCIA :DRAGON :CIUDAD<50 THEN
  CIUDAD. DESTRUIDA STOP
  IF DISTANCIA :DRAGON :CIUDAD1<50 THEN
  CIUDAD.DESTRUIDA STOP
  MOVIMIENTO.CABALLERO
  IF ACERTADO? THEN DRAGON.DESTRUIDO
  STOP
  JUGAR
END

TO MOVIMIENTO.DRAGON
  TELL :DRAGON
  MOVPALANCA JOYSTICK 1
  FD 10
END

TO MOVPALANCA :DIR
  IF :DIR<0 STOP
  SETH :DIR*45
END

TO DISTANCIA :A :B
  TELL :A
  MAKE"X1 XCOR
  MAKE"Y1 YCOR
```



SPRITES, TS?, que devuelve "TRUE si hay un sprite tocando al sprite en curso. ACERTADO? pone al dragón como sprite en curso y después pregunta si hay algo tocándolo.

La instrucción JOYSTICK toma 0 o 1 como entrada (que corresponden a las puertas 1 y 2). La salida es −1 si la palanca de mando está en el centro, 0 si está arriba, 1 si está en 45°, 2 si está en 90°, y así sucesivamente hasta 7. Aquí simplemente establecemos el encabezamiento del dragón en 45° multiplicado por el número de salida.

Mediante la utilización de sprites es fácil conseguir explosiones y efectos similares. Encima del objeto a destruir aparece de modo intermitente una forma que representa la explosión. Al sprite LLAMA le damos un número bajo para que tenga alta prioridad y aparezca encima de los otros sprites.

El ordenador controla al caballero, pero emplea una estrategia defensiva muy simple: el caballero se encabeza directamente hacia el dragón. Tal como está el juego, el monstruo puede escabullirse por el lado del caballero y causar estragos.

¿Cómo podemos mejorar la estrategia defensiva del caballero? Una forma sencilla consiste en aumentar su velocidad; con sólo incrementarla de 10 a 11 al dragón le es mucho más difícil poder escabullirse. (¡Salir de la pantalla es hacer trampa!). Una estrategia acertada sería cortarle el paso al dragón orientando su encabezamiento hacia la línea entre éste y la ciudad y quedándose allí.

Con tres insectos

Los insectos parten de las esquinas de un triángulo:

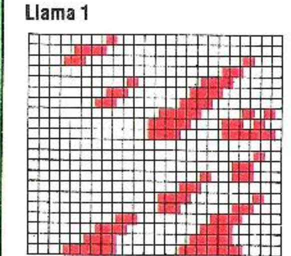
```
TO INSECTOS
   PREPARACION MOVER.INSECTOS
END
TO PREPARACION
   DRAW FULLSCREEN TELL 0 HT PU SETXY
   (−100)(−100) TRI 200 POSICION 1 (−100)
   (−100) POSICION 2 0 73 POSICION 3 100
   (−100)
END
TO TRI :LADO
   PD REPEAT 3 [FD :LADO RT 120] PU
END
TO POSICION :NUM :X :Y
   TELL :NUM SETSHAPE 3 PU SETXY :X :Y PD ST
END
TO MOVER.INSECTOS
   SEGUIR 1 2 SEGUIR2 3 SEGUIR 3 1
   MOVER.INSECTOS
END
TO SEGUIR :A :B
   TELL :B MAKE"X XCOR MAKE "Y YCOR TELL :A
   SETH TOWARDS :X :Y FD 10
END
```

Editor de sprites

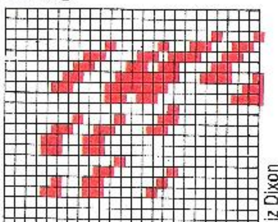
```
   TELL :B
   MAKE"X2 XCOR
   MAKE"Y2 YCOR
   OUTPUT SQRT((:X1 − :X2)*(:X1 −:X2)+(:Y1
   − :Y2)*(:Y1 −:Y2))
END
TO CIUDAD.DESTRUIDA
   TELL :CIUDAD
MAKE"X XCOR
   MAKE"Y YCOR
   FLASH :X :Y :NARANJA
   TELL :CIUDAD1
   MAKE"X2 XCOR
   MAKE"Y2 YCOR
   FLASH1 :X2 :Y2 :NARANJA
   ESCONDER.SPRITE
   SPLITSCREEN
   REPEAT 3 [PRINT"]
   PRINT [DESTRUIDA LA CIUDAD!]
END
TO FLASH :X :Y :COLOR
   TELL :LLAMA
   PENCOLOR :COLOR
   SETXY :X :Y
   ST
   REPEAT 6 [SMALLX SMALLY ESPERAR BIGX
   BIGY ESPERAR]
END
TO FLASH1 :X2 :Y2 :COLOR
   TELL :LLAMA1 PENCOLOR :COLOR
SETXY :X2 :Y2
   ST REPEAT 6 [SMALLX SMALLY ESPERAR BIGX
      BIGY ESPERAR]
END
TO MOVIMIENTO.CABALLERO
   TELL :DRAGON
   MAKE"X XCOR
   MAKE"Y YCOR
   TELL :CABALLERO
   SETH TOWARDS :X :Y
   FD 10
END
TO ACERTADO?
   TELL :DRAGON
   IF TS? THEN OUTPUT "TRUE
   OUTPUT "FALSE
END
TO DRAGON.DESTRUIDO
   TELL :DRAGON
   MAKE "X XCOR
   MAKE "Y YCOR
   FLASH :X :Y :NEGRO
   ESCONDER.SPRITE
   SPLITSCREEN
   REPEAT 3[PRINT"]
   (PRINT[DRAGON MUERTO A UNA DISTANCIA
      DE] DISTANCIA :DRAGON :CIUDAD)
END
TO ESCONDER.SPRITE
   TELL :LLAMA HT
   TELL :LLAMA1 HT
   TELL :CIUDAD HT
   TELL :CIUDAD1 HT
END
TO ESPERAR
   REPEAT 100 []
END
```



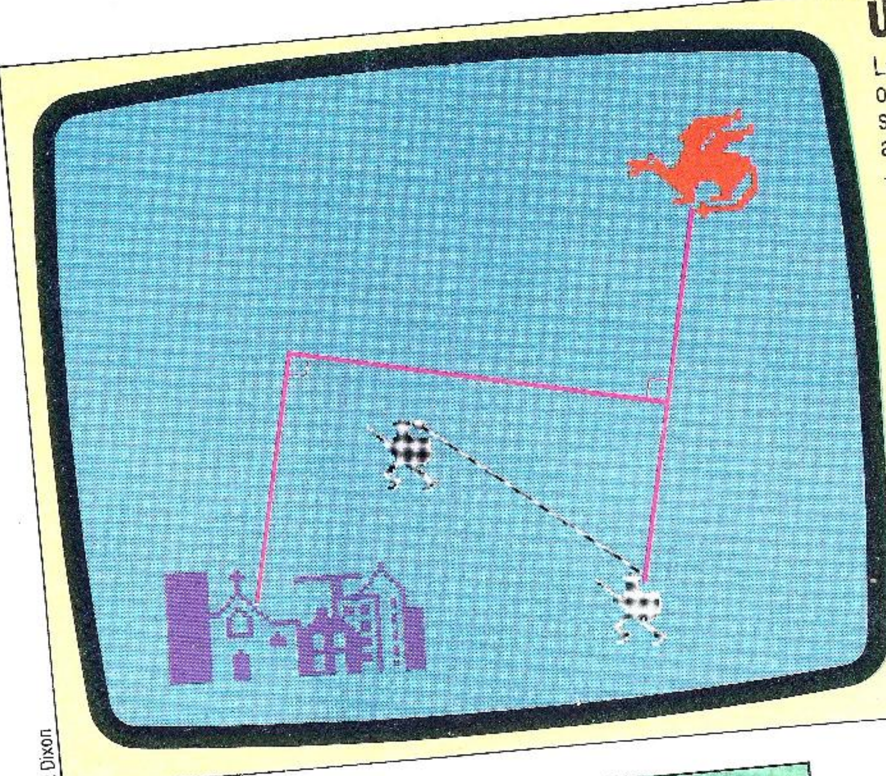
Llama 1



Llama 2



Liz Dixon



Una estrategia mejor

La mejor estrategia (para ambas partes) es orientarse hacia el lugar más próximo a la ciudad sobre la bisectriz perpendicular de la línea que une al caballero y al dragón (véase diagrama):

```
TO MOVER.CABALLERO
  TELL :DRAGON MAKE "DX XCOR MAKE"DY
  YCOR TELL :CABALLERO MAKE"KX XCOR
  MAKE "KY YCOR TELL :CIUDAD MAKE"CX
  XCOR MAKE "CY YCOR
  MAKE"SX(:DX+:KX)/2
  MAKE"SY(:DY+:KY)/2
  MAKE"VX(:DY-:KY)
  MAKE"VY(:KX-:DX)
  MAKE"FACT(:VX*(:CD-:SX)+:VY*
  (:CY-:SY))/((:VX*:VX)*(:VX*:VX)+
  (:VY*:VY)*(:VY*:VY))
  MAKE"X :SX+:FACT*:VX
  MAKE"Y :SY+:FACT*:VY
  TELL :CABALLERO SETH TOWARDS :X :Y
  FD 10
END
```

Movimiento acertado

Un movimiento mejorado para el caballero, en el cual éste se orienta para interceptar al dragón:

```
TO MOVIMIENTO.CABALLERO
  TELL :DRAGON MAKE"X XCOR MAKE "Y
  YCOR
  TELL :CIUDAD
  MAKE"DIRIGIENDOSE TOWARDS :X :Y
  TELL :CABALLERO SETH 270+:DIRIGIENDOSE
  IF XCOR< :X THEN LEFT 180
  FD 10
END
```

Control por teclado

Control por teclado del dragón:

```
TO MOVIMIENTO.DRAGON
  TELL :DRAGON MOVER TECLALEIDA FD 10
END

TO MOVER :DIR
  IF :DIR="W THEN SETH 0
  IF :DIR="S THEN SETH 90
  IF :DIR="Z THEN SETH 180
  IF :DIR="A THEN SETH 270
END

TO TECLALEIDA
  IF RC? THEN OUTPUT READCHARACTER
  OUTPUT"
END
```



Abreviaturas

BACKGROUND	BG
PENCOLOR	PC

Complementos al LOGO

El LOGO Spectrum y el LOGO Apple no incorporan gráficos sprite.

Los usuarios de Atari deben tomar nota de estas diferencias:

1) TS? no existe. Omita la línea IF ACERTADO? etc. de JUGAR e inserte la siguiente línea como la última de PREPARAR.PANTALLA:

```
WHEN TOUCHING :DRAGON :CABALLERO
[DRAGON.DESTRUIDO STOP]
```

2) No hay equivalentes para BIGX, BIGY, SMALLX y SMALLY. Omítalas.

3) Para BACKGROUND utilice SETBG, y para PENCOLOR emplee SETPC. Los códigos de color son diferentes.

4) Lo más sorprendente es que TOWARDS no existe en el LOGO Atari (sí se incluye en las versiones LCSI para el Apple y el Spectrum). Por lo tanto, reemplace las líneas:

```
SETH TOWARDS :X :Y
FD 10
```

de SEGUIR y MOVIMIENTO.CABALLERO por:

```
MAKE"FRAC 10/(SQRT((XCOR-:X)*(XCOR
-:X)+(YCOR-:Y)*(YCOR-:Y)))
SETPOS LIST (XCOR+(:X-XCOR)*:FRAC)
(YCOR+(:Y-YCOR)*:FRAC)
```



En suspenso

Vamos a estudiar detalladamente el mecanismo de las interrupciones al que repetidamente nos venimos refiriendo

Una conocida aplicación de las interrupciones es la de espera de la pulsación de una tecla del teclado. Si un programa accediera directamente a éste —que suele hacerse a través del sistema operativo— para obtener el siguiente carácter de entrada, entonces cualquier tecla pulsada cuando el programa estuviera haciendo otra cosa quedaría sin efecto. Incluso cuando el procesador se aplica de lleno a procesar la entrada del teclado, es posible que pierda algún carácter, en especial el que siga al carácter que requiera un proceso extra, como es el retorno de carro.

El teclado encuentra la solución interrumpiendo al procesador en cuanto se oprime una tecla, de modo que éste detiene lo que está haciendo y ejecuta la "rutina de atención de interrupción". Tal rutina toma el carácter que acaba de introducirse y lo coloca en una sección de memoria reservada con el nombre de *buffer del teclado*. Seguidamente el procesador puede volver a lo que estaba haciendo como si nada hubiera ocurrido.

Siempre que se llama a la rutina de entrada de teclado del sistema operativo, ésta no inspecciona directamente el teclado sino que toma el carácter siguiente del buffer (si el buffer está vacío esperará a que aparezca un carácter). Este mecanismo permite al usuario "adelantarse" en el tecleado respecto a lo que aparece en pantalla, y a la vez asegura que no se pierda ningún carácter.

Hay, sin embargo, dos problemas posibles. El usuario puede teclear tan rápido que el buffer se llene más aprisa de lo que el programa tarda en tratar las entradas, produciendo su desbordamiento (*overflow*). Esto se resuelve con un compromiso en el tamaño del buffer, de tal forma que no se produzcan desbordamientos y no se derroche un excesivo espacio de memoria, que tan valioso puede resultar. El segundo problema surge con aquellos usuarios que se sienten incómodos cuando un carácter no aparece en la pantalla inmediatamente después de haber oprimido una tecla. Puede que entonces continúen oprimiéndola y, de resultas, generen docenas de caracteres que van a parar al buffer, hasta desbordarlo de nuevo. Un problema que se resuelve familiarizándose con el ordenador.

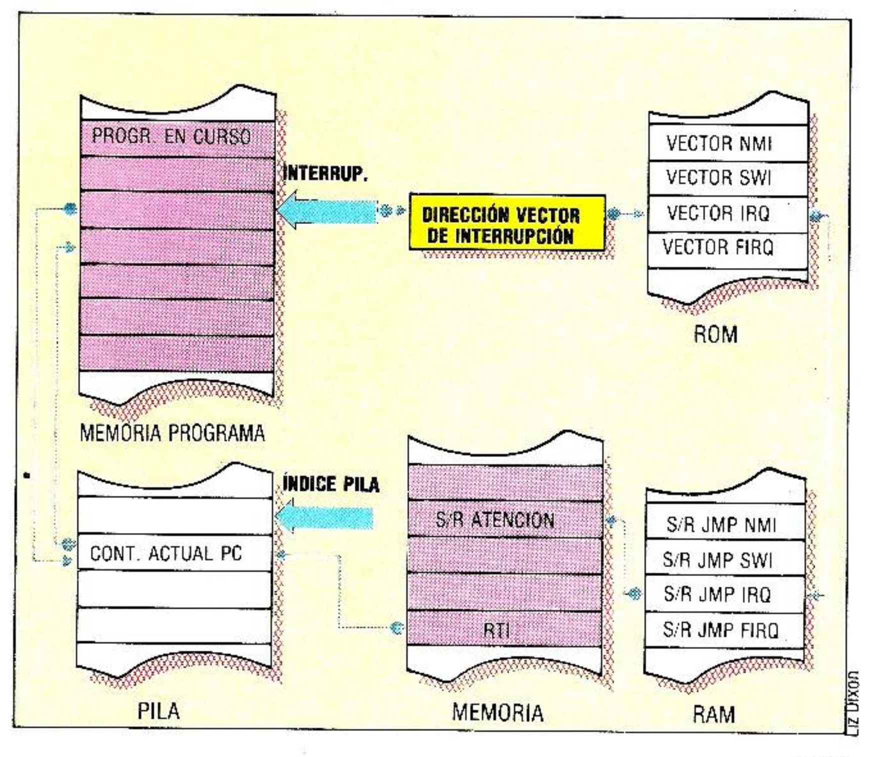
Otra aplicación útil de las interrupciones sucede cuando hay una salida hacia la impresora, que es con frecuencia una de las operaciones ejecutadas por el micro que más tiempo requieren. Durante la impresión, puede que se le exija al procesador trabajar durante 100 microsegundos mientras envía un carácter a la impresora y esperar después miles de microsegundos hasta que la impresora procese ese carácter. Para obviar la dificultad se recurre a un sistema de *spooling* (de SPOOL: *Simultaneous Peripheral Operation On Line*: operación simultánea de periféricos en línea). Éste sitúa en una cola los archivos a imprimir, y parte del primer archivo que está en la cola es almacenada en otra área del buffer de la memoria. La puerta que está al servicio de la impresora interrumpirá al procesador en cuanto la impresora esté preparada para la recepción de otro carácter. La rutina de atención de interrupción enviará entonces el siguiente carácter desde el buffer, o (si el buffer está vacío) cargará la siguiente sección del fichero que encabeza la cola, llevándola al buffer. De este modo, la impresora puede continuar su trabajo entre bastidores, mientras el procesador queda libre para atender cualquier otra cosa.

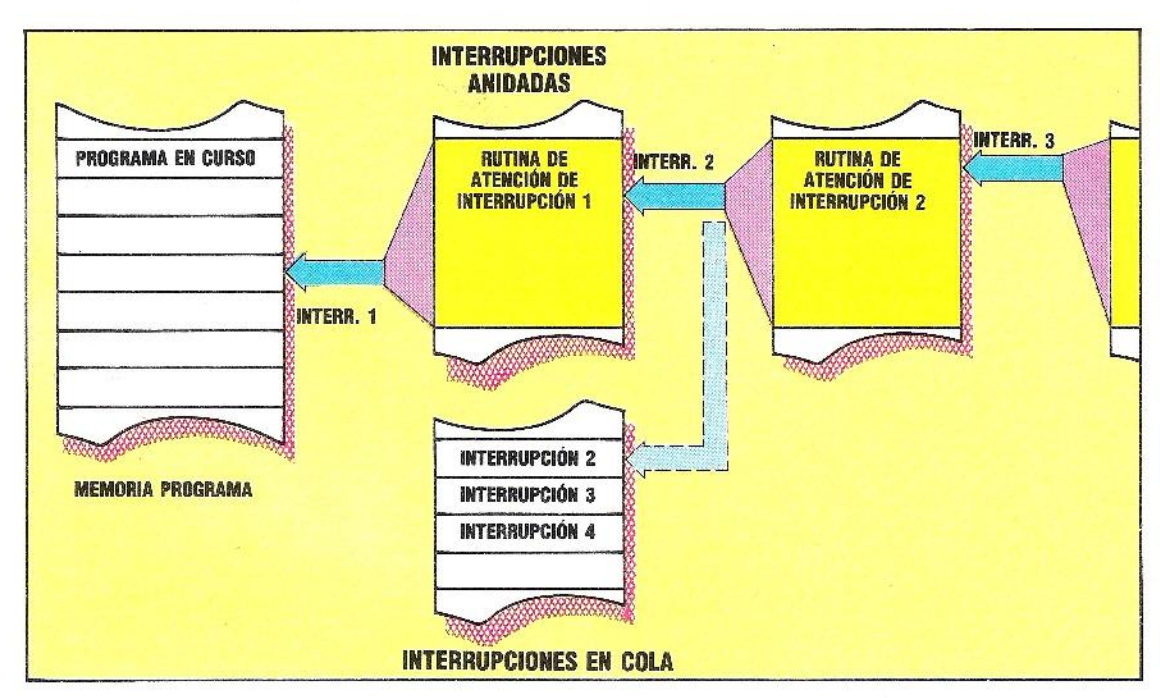
Tipos de interrupción

Existen operaciones realizadas por el procesador (el acceso a discos es una de ellas) en las que una interrupción puede causar pérdida de datos o cualquier otro incidente. Debe haber, por tanto, un mecanismo para "enmascarar" interrupciones de modo que el procesador pase por alto las que puedan ocurrir durante una operación particularmente delicada. En este caso, es aconsejable anotar que ha habido una interrupción y que será tratada más adelante.

Por otro lado, si estamos manejando una interface de discos llevada por medio de interrupciones, éstas deben tener prioridad y no han de enmascararse bajo ningún pretexto: es lo que ha motivado el concepto de interrupción *no enmascarable*. Tal

Interrupción número uno
Al ocurrir una interrupción el procesador concluye la ejecución de la instrucción en curso y guarda en la pila el contenido actual del contador de programa. La dirección vector de la interrupción adecuada se carga en dicho contador, pasándose el control a dicha dirección, que suele encontrarse en la ROM. Esta dirección señala a su vez a otra de la RAM, donde la instrucción JMP dirige el control a la rutina de atención de interrupción requerida. La cual concluye con una instrucción RTI para devolver el control al programa principal por medio de la dirección de retorno guardada en la pila. Dado que la instrucción JMP está almacenada en la RAM, el programador puede encontrarla y alterarla de tal modo que el control pueda ser pasado primero a una rutina de usuario con finalidad específica y después a la rutina de atención habitual





Interrupciones interrumpidas
Si durante una interrupción tuviera lugar otra interrupción, el procesador puede optar por "anidar" las interrupciones: cada vez que se da una interrupción, el PC es salvado en una pila para ocuparse de la nueva interrupción inmediatamente y devolver después el control a la dirección que se salvó en la pila. Los límites de estos anidamientos dependen de la capacidad de las pilas y de la posibilidad de los dispositivos generadores de interrupciones para soportar demoras en el proceso de sus interrupciones.
Otra opción del procesador consiste en apilar los detalles de toda interrupción en una cola de interrupciones. Concluida la primera interrupción, el procesador inspecciona la cola y procesa la interrupción que encuentre allí por orden hasta que la cola se vacía pasando después el control al programa

interrupción puede proceder de un circuito que detecte una caída en la tensión de la corriente eléctrica: su rutina de atención debe comenzar inmediatamente a salvar el estado de la tarea actual mientras quede electricidad.

Si las interrupciones pueden proceder de más de una fuente, tendremos que considerar la idea de interrupciones *anidadas*. Si sucede una interrupción mientras el procesador está atendiendo otra interrupción, hay dos posibles estrategias para su tratamiento. Primero, que se deseche la nueva interrupción mientras no se complete la que está en curso. Segundo, que se jerarquicen las interrupciones según un criterio de urgencia, de tal modo que una interrupción de máxima prioridad pueda detener el tratamiento de otra con menor prioridad. En este caso el sistema operativo ha de saber tratar el anidamiento de rutinas de atención de interrupciones.

Interrupciones de software

La instrucción SWI, mencionada brevemente en la página 1057, puede ser empleada en un programa para volver convenientemente al sistema operativo mediante la generación de su propia interrupción, llamada *interrupción de software* (para distinguirla de las que genera el hardware y de las que hasta ahora hemos estado hablando). Pero también podemos utilizar instrucciones SWI para que actúen como puntos de discontinuidad o ruptura en un programa en lenguaje máquina como ayuda a la depuración de errores; esta facilidad la procuran muchos monitores de código máquina basados en ROM, así como los paquetes de depuración. El usuario escoge puntos en la codificación donde hará una pausa la ejecución del programa, y las instrucciones colocadas en estas posiciones son reemplazadas por las SWI. Cuando se ejecuta el programa, la rutina de atención de interrupciones permite al programador inspeccionar y, si es el caso, alterar el contenido de los registros y las posiciones de la memoria, y ver lo que el programa está haciendo exactamente. Reanudada la ejecución, el monitor-depurador sustituye la instrucción indicada por el punto de ruptura SWI para que continúe el programa desde ese mismo punto.

El 6809 tiene tres mecanismos de interrupción independientes: $\overline{IRQ}$ (Interrupción ReQuerida), $\overline{FIRQ}$ (Primera —*First*— Interrupción ReQuerida) y NMI (Interrupción No enMascarable). Las tres son activadas por una señal apropiada que se recibe en tres patillas del chip del procesador. La barra sobre el nombre (p. ej., en $\overline{IRQ}$) sirve para indicar que son activadas por una señal 0 en el procesador, en lugar de una señal 1. Estas tres patillas se conectan al bus principal para que los chips periféricos tales como el 6820 y el 6850 puedan conectar sus patillas de salida de interrupción requerida a las mismas líneas del bus. Cuando se programan los chips, las interrupciones pueden ser permitidas y el envío de las señales apropiadas puede ser automático.

Hay también tres interrupciones de software causadas por las instrucciones SWI; SW12 y SW13.

Cuando se produce una interrupción, el control se pasa a la dirección de *vector* contenida en una posición específica en la parte superior de la memoria. Estas direcciones de vector se encuentran generalmente en la ROM, por lo que el control pasa desde allí hasta la misma dirección fijada. Pero esta dirección a su vez suele encontrarse en la RAM y contendrá una instrucción JMP, por cuyo motivo puede cambiarse el destino final hacia la rutina de atención propia del usuario. Las posiciones de memoria son éstas:

Tipo de interrupción	Vector
$\overline{NMI}$	$FFFC
SWI	$FFFA
$\overline{IRQ}$	$FFF8
$\overline{FIRQ}$	$FFF6
SW12	$FFF4
SW13	$FFF2

Es conveniente observar también que los dos bytes superiores de la memoria (el $FFFE y el $FFFF) con-

tienen el *vector de reinicialización*, la dirección a la que se transfiere el control al dar la corriente eléctrica o reinicializar el hardware; suele tratarse de la dirección de inicio del monitor de la ROM. Asimismo, los dos bytes situados en $FFF0 y $FFF1 los reserva Motorola para un posible empleo futuro.

La información sobre interrupciones se halla en tres bits del registro de código de condición (CC): el bit 4 (I), el bit 6 (F) y el bit 7 (E). Si el bit I se pone a uno se enmascara la interrupción IRQ, si el que se pone a uno es el bit F enmascaramos la FIRQ. El bit E sirve al procesador para distinguir entre IRQ o NMI y FIRQ: si E se pone a uno es que ha ocurrido una IRQ o NMI, y si se pone a cero, una FIRQ. Cuando se recibe una interrupción, ésta suele tener el mismo tratamiento que una llamada de subrutina: los contenidos de algunos o todos los registros son llevados a una pila para permitir que el control pueda retornar al mismo punto por donde iba la ejecución del programa. La rutina de servicio de interrupción acaba con una instrucción RTI (parecida a una RTS), que tiene la virtud de restaurar los registros de la pila y devolver el control al programa original.

La diferencia entre la FIRQ y las otras dos interrupciones reside en que FIRQ sólo guarda en la pila el registro contador del programa (PC) y el de código de condición (CC), resultando así mucho más rápida operativamente que las demás. La rutina de atención de interrupciones debe, sin embargo, restaurar cualquier registro empleado, lo que desaconseja este tipo de interrupción en rutinas que utilicen más de uno o dos registros. Estamos ahora en condiciones de ver dónde se emplea el bit E, dado que se emplea la misma instrucción RTI para concluir IRQ y FIRQ, aunque el procesador debe determinar qué registros han de ser restaurados de la pila. La secuencia de operaciones es la siguiente:

1) Se ejecuta la instrucción en curso.
2) El bit I se pone a uno, impidiendo más interrupciones IRQ. Si se trata de una interrupción FIRQ o NMI entonces también se pone a uno el bit F para inhibir la FIRQ. Aunque SW12 y SW13 no enmascaran otras interrupciones, la SW1 sí lo hace.
3) En caso de una FIRQ el bit E se pone a cero, de lo contrario se pone a uno.
4) El vector colocado en las posiciones apropiadas de memoria es cargado en el PC y la ejecución continúa a partir de esa dirección.

Nuestro primer programa propone otro útil empleo de las interrupciones: el mantenimiento de un reloj de tiempo real. Vamos a suponer que algún tipo de dispositivo temporizador (p. ej., un chip de propósito especial, como el temporizador 6840) se ha conectado al PIA en la posición $5000. La primera subrutina permitirá las interrupciones y establecerá en $50 un contador de 16 bits. La rutina de atención de interrupciones se limitará a incrementar el contador para que en cualquier momento la inspección de $50 proporcione el número de señales de temporizado que se han recibido y a partir de las cuales se puede calcular el tiempo si se conoce el tiempo inicial y la frecuencia de dichas señales.

El segundo programa ejemplo supone la conexión de una impresora al mismo PIA en $E000. Emplearemos un buffer de longitud indefinida en la posición $100 para guardar en él una línea de salida que será impresa por la rutina de atención. En $50 se pone un flag a cero mientras se está imprimiendo la línea, y a uno al concluir su impresión. Esto último permitirá la introducción de otra rutina para volver a llenar el buffer y de la que de momento no nos ocupamos aquí. $51 y $52 contienen un apuntador al buffer que proporciona la dirección del siguiente carácter a imprimir. La primera subrutina establece el PIA, el flag y el apuntador al buffer preparados para una nueva línea.

Programa núm. 1

```
PIACR    EQU    $E001        Registro de control del PIA
PIADR    EQU    $E000        Registro de datos del PIA
INTRP    EQU    $2000
CLK1     EQU    $50
CLK2     EQU    $51
         ORG    $1000        Subrutina que inicializa el reloj
INITCK   CLR    CLK1         Limpia posiciones del reloj
         CLR    CLK2
         LDA    #%00000101   Posibilita interrupciones del PIA
         STA    PIACR
         ANDCC  #%11101111   Posibilita la IRQ
WAITCK   TST    CLK2         Espera el primer incremento
         BEQ    WAITCK
         RTS
         ORG    INTRP        Rutina de atencion de la interr.
         LDA    PIADA        Borra la interrupcion
         LDD    CLK1         Toma el contador
         ADDD   #1           Incrementa el contador
         STD    CLK1
         RT1
```

Programa núm. 2

```
PIACR    EQU    $E001        Registro de control del PIA
PIADR    EQU    $E000        Registro de datos del PIA
INTRP    EQU    $2000
CR       EQU    13           Retorno de carro
BUFFER   EQU    $100         Direccion del buffer
BUFPTR   EQU    $51          Apuntador del buffer
FLAG     EQU    $50          Flag de fin de linea
         ORG    $1000        Subrutina para la prep. global
         CLR    FLAG         Borra el flag de fin de linea
         LDX    #BUFFER      Inicializa el apuntador del buffer al
                             comienzo de este
         STX    BUFPTR
         CLR    PIACR        Registro de direccion de datos
         LDA    $FF          Prepara lineas para la salida
         STA    PIADR
         LDA    #%00000101   Posibilita interrupciones del PIA
         STA    PIACR
         ANDCC  #%11101111   Posibilita la IRQ
         RTS
         ORG    INTRP        Rutina de atencion interr.
         LDX    BUFPTR       Apuntador del buffer
         LDA    ,X+          Toma del buffer el sig. car.
         STA    PIADR        Lo imprime
         LDB    PIADR        Borra la interrupcion
         STX    BUFPTR       Apuntador del buffer incrementado
         CMPA   #CR          Fue fin de linea?
         BNE    FINISH       Salto si no fue asi
         INC    FLAG         Activa el flag, si fue asi
FINISH   RTI
```

Para el Commodore

Esta vez desarrollaremos para el Commodore 64 un programa de utilidad para búsqueda de variables, tal como lo hiciéramos para el BBC Micro y el Spectrum (pp. 1144-1145)

En la versión Commodore del programa se deben abreviar muchos nombres de variables para evitar que incluyan una palabra clave de BASIC. Por ejemplo, NEWLINE (nueva línea) no se puede utilizar como nombre de una variable porque comienza con NEW, y TEXTPOINTER (apuntador de texto) tampoco se puede emplear porque incluye INT.

Los cambios en la parte inicial del programa son necesarios debido a las diferencias en cuanto a la forma en que se almacena una línea de BASIC en la memoria del ordenador. En el BBC Micro y el Spectrum, en el formato interno una línea de BASIC empieza con un número de línea de dos bytes, con el byte más significativo primero, y uno o dos bytes para la longitud de la línea. En el Commodore 64, una línea de BASIC comienza con un apuntador de dos bytes que indica el comienzo de la línea siguiente, y un número de línea de dos bytes, con el byte menos significativo primero en ambos casos.

Aún debemos saltar las líneas REM y las series entre comillas, pero no es necesario buscar ningún otro caso especial que pudiera causar confusión, como los números hexadecimales en el BBC o la forma binaria oculta de números en el Spectrum.

Utilidad de búsqueda para el C64

```
30000 INPUT "NOMBRE A BUSCAR";T$
30010 RE=143
30020 COMILLA=34
30030 NL=0
30040 APNTEXTO=2049
30050 SIGLINEA=PEEK(APNTEXTO)+256*PEEK (APNTEXTO+1)
30070 APNTEXTO=APNTEXTO+2
30080 NUMLINEA-PEEK(APNTEXTO)+256*PEEK(APNTEXTO+1)
30085 IF NUMLINEA>-30000 THEN END
30090 APNTEXTO-APNTEXTO+2
30140 IF PEEK(APNTEXTO)=NL THEN APNTEXTO=APNTEXTO+1:GOTO 30050
30150 IF PEEK(APNTEXTO)<>RE THEN GOTO 30180
30160 REM SALTAR LINEA REM
30170 APNTEXTO=SIGLINEA:GOTO 30050
30180 IF PEEK (APNTEXTO)<>COMILLA THEN GOTO 30300
30190 REM SALTAR TODO LO QUE ESTE ENTRE COMILLAS, PARARSE AL FINAL DE LA LINEA EN CASO DE COMILLA SIN
      PAREJA
30200 APNTEXTO-APNTEXTO+1
30210 IF PEEK(APNTEXTO)-NL THEN APNTEXTO-APNTEXTO+1:GOTO 30500
30220 IF PEEK(APNTEXTO)<>COMILLA THEN GOTO 30200
30230 APNTEXTO=APNTEXTO+1
30235 GOTO30140
30290 REM EL PRIMER CARACTER DEL NOMBRE DEBE SER UNA LETRA
30300 IF PEEK(APNTEXTO)>=ASC("A") AND PEEK (APNTEXTO)<=ASC("Z") THEN GOTO 30330
30310 APNTEXTO-APNTEXTO+1
30320 GOTO30140
30330 NOMBRE$-""
30340 NOMBRE$-NOMBRE$+CHR$(PEEK(APNTEXTO))
30350 APNTEXTO=APNTEXTO+1
30360 REM LETRA O DIGITO DESPUES DEL PRIMER CARACTER
30370 IF PEEK(APNTEXTO)>-ASC("A") AND PEEK(APNTEXTO)<-ASC("Z") THEN GOTO 30340
30390 IF PEEK(APNTEXTO)>=ASC("0") AND PEEK(APNTEXTO)<=ASC("9") THEN GOTO 30340
30410 REM FINAL CON $ PARA VARIABLE EN SERIE, % PARA VARIABLE ENTERA
30420 IF PEEK(APNTEXTO)=ASC("$") THEN NOMBRE$-NOMBRE$+"$":APNTEXTO=APNTEXTO+1:GOTO 30450
30430 IF PEEK(APNTEXTO)-ASC("%") THEN NOMBRE$=NOMBRE$+"%":APNTEXTO=APNTEXTO+1
30440 REM (SI MATRIZ O FUNCION
30450 IF PEEK(APNTEXTO)-ASC("(") THEN NOMBRE$=NOMBRE$+"(":APNTEXTO=APNTEXTO+1
30460 IF NOMBRE$=T$ THEN PRINT NOMBRE$;"  EN LA LINEA";NUMLINEA
30470 GOTO 30140
30480 END
```

La sección del programa que realmente extrae los nombres de variables busca primero una letra del alfabeto, luego letras o dígitos, y finalmente busca un signo $ o % que indique una variable en serie o entera y un carácter (, que indicaría una función o matriz. El Commodore 64 no admite el carácter de subrayado que sí se puede incluir en los nombres de variables en el BBC Micro y el Spectrum.

Si bien el Commodore 64 puede visualizar tanto letras en mayúscula como en minúscula, la diferencia sólo existe en cuanto a la forma del carácter que aparece en la pantalla, y no en el código interno para el carácter. Por tanto, el programa sólo ha de buscar las mayúsculas en el nombre de una variable.

La versión del programa para el Commodore 64 se utiliza de la misma manera que las versiones para el BBC Micro y el Spectrum. Entre el programa de búsqueda y guárdelo (SAVE), luego cargue (LOAD) el programa en el cual se realizará la búsqueda y añádale el programa de búsqueda. Entonces ya puede buscar en el programa mediante "RUN 30000" y tecleando el nombre de la variable cuando el programa se lo solicite, terminando el nombre con "(" si desea hallar el nombre de una matriz.

Existe un procedimiento sencillo para unir dos programas guardados en el Commodore 64, siempre y cuando los números de líneas del primer programa sean todos inferiores a los números de línea del segundo programa. El método utiliza dos de los apuntadores de la página cero: TXTTAB, en las direcciones 43 y 44, que retiene la dirección donde comienza el programa en BASIC, y VARTAB, en las direcciones 45 y 46. Un programa en BASIC acaba con un byte conteniendo cero que indica el final de la última línea del programa, luego otros dos bytes cero que marcan el final del programa. La dirección en VARTAB normalmente es el byte que sigue al último de estos ceros. Para unir los dos programas, primero cargue (LOAD) el programa que tenga los números de línea inferiores, después digite:

```
PRINT PEEK(45), PEEK(46)
```

Si el primer número está comprendido entre 2 y 255, réstele 2 y coloque (POKE) el resultado en la dirección 43. Si es cero o uno, digite POKE 254 o 255 en la dirección 43 y coloque (POKE) en la dirección 44 uno menos que el resultado de PEEK(46). Entonces puede cargar el segundo programa, y finalmente debe digitar:

```
POKE 43,1: POKE 44,8
```

Esto volverá a colocar el valor normal en el apuntador de "comienzo de BASIC", y ahora los programas se unirán entre sí.

GRAN SORTEO
MI COMPUTER
VALIDO UNICAMENTE EN VENEZUELA

10 Computadores Portátiles Data General/One y 4 Becas para estudiar computación en el Instituto de Computación.

C CENESCO

Coleccione Mi Computer ...y gánese el suyo!

Entre ganando en el fascinante mundo de la Informática con Mi Computer, que sortea 10 fabulosos computadores personales Data General/One valorados en Bs. 60.000,00 c/u. con el respaldo de CENESCO, además de 4 Becas con un costo de Bs. 13.000,00 c/u., para estudiar la carrera del futuro: Computación en el Instituto de Computación CENESCO.

Participar es fácil...!

BASES:

1. Llene el cupón que encontrará en este fascículo.
2. Recorte las esquinas numeradas del 56 al 65 que aparecen en las contraportadas de los fascículos del 56 al 65 y péguelas todas en su lugar correspondiente en el cupón.
3. Envíe antes del 27 de octubre de 1986 el cupón al apartado de correos No. 51285 Z. P. 1050, y participe gratis en el sorteo de un computador, que se realizará el 10 de noviembre de 1986 y cuyos resultados aparecerán en el Diario "El Universal" del 17 de noviembre de 1986. Al ganador se le notificará por carta y se le hará entrega en la ciudad de Caracas.
4. Otros 2 computadores se sortearán con los cupones que aparecerán en los fascículos 65-75.

Todos los sorteos contarán con la supervisión del Ministerio de Fomento y con la garantía de PLANETA

El equipo incluye: Micro Computador DATA GENERAL/ONE 256 Kb de memoria principal. Unidad de Diskette 720 Kb y el siguiente software: Sistema Operativo MS-DOS, Lenguaje de Programación GW-BASIC, Diskette con curso instructivo, Interface RS-232-C, Interface RS-422-C y Maletín para Micro


59